НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут телекомунікаційних систем

Кафедра Телекомунікацій

До захисту допущено:

Завідувач кафедри

_____ Сергій КРАВЧУК

«___»_____2021 р.

Дипломна робота

на здобуття ступеня <u>бакалавра</u>

<u>Спеціальності 172 «Телекомунікації та радіотехніка»</u>

на тему: «Побудова кластерної мережі SDN центру обробки даних з використанням HPE VAN SDN Controller**»**

Виконав: студент ____4____ курсу, групи _____ТЗ-72_____
(шифр групи)

_____Марінов Антон Ігорович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник _____доктор технічних наук, професор Романов О.І._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент ___кандидат технічних наук, доцент Созонник Г.Д._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем

Кафедра Телекомунікацій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій КРАВЧУК

«___»_____2021 р.

ЗАВДАННЯ

на дипломну роботу студенту

Марінову Антону Ігоровичу

1. Тема роботи : «Побудова кластерної мережі SDN центру обробки даних з використанням HPE VAN SDN Controller»

Керівник роботи професор, д.т.н. Романов Олександр Іванович

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 14 квітня 2021 р. № 1007-с

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи:

- технології побудови макету мережі – SDN, кластеризація, віртуалізація,

- програмний емулятор для побудови та запуску мережі – Mininet;

- програмний продукт для керування, створення та запуску віртуальних машин – VirtualBox;

- спосіб побудови мережі у Mininet - побудова мережі з віддаленим контролером та мережевими пристроями, де контролер буде працювати на тій же віртуальній машині, де запущена мережа;

- віддалений віртуальний контролер – HPE VAN SDN Controller;

- мова програмування для написання необхідних топологій мережі та запуску окремих додатків – Python;

- протоколи передачі даних при наведені прикладів – ICMP, UDP, TCP;

- зв’язок присутній між мережевими елементами, що знаходяться в одному кластері (підключені до одного й того ж комутатору).

4. Зміст роботи:

1. Загальний огляд кластерних мереж, технології кластеризації;
2. Система ЦОД. Принципи побудови кластерної ЦОД мережі;
3. Принципи впровадження технології віртуалізації, SDN до мережі ЦОД;
4. Технологія SDN. Мережеві елементи та її особливості у роботі;
5. Призначення, можливості а також переваги і недоліки програмного емулятору Mininet;
6. Огляд мережевого контролеру HPE VAN SDN та особливості в роботі. Приклади використання на Mininet;
7. Розробка та побудова практичного макету на базі SDN мережі у Mininet;
8. Підбиття підсумків щодо проробленої роботи.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) :

Слайд № 1 «»

Слайд № 2 «»

Слайд № 3 «»

Слайд № 4 «»

Слайд № 5 «»

Слайд № 6 «»

Слайд № 7 «»

6. Дата видачі завдання – _____05.02.2021

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Визначення з темою, метою дослідження та пошук літературних джерел.	08.02.2021 – 18.03.2021	Виконано
2	Створення структури роботи та завершення аналізу знайденої літератури.	22.03.2021 – 26.03.2021	Виконано
3	Написання першої частини дипломної роботи: “Аналіз принципів побудови центрів обробки даних”. Малювання ілюстрацій для роботи.	29.03.2020 – 02.04.2021	Виконано
4	Написання другої частини дипломної роботи: “Принцип роботи кластерної мережі. Технології SDN, віртуалізації та кластеризації у мережах. Шляхи побудови кластерної мережі центру обробки даних на базі SDN”.	05.04.2021 – 14.04.2021	Виконано
5	Написання третьої частини дипломної роботи: “Побудова моделі кластерної SDN мережі ЦОД на базі платформи Mininet”. Встановлення необхідних програмних компонентів та їх налаштування для виконання практичної частини.	19.04.2021 – 30.04.2021	Виконано
6	Виконання практичної частини та написання четвертої частини дипломної роботи: “Дослідження та побудова мережевого макету кластерної SDN мережі ЦОД на базі платформи Mininet”	05.05.2021 – 14.05.2021	Виконано
8	Оформлення частин диплому. Розбиття інформації по розділам. Створення необхідних ілюстрацій.	17.05.2021 – 25.05.2021	Виконано
9	Написання вступної частини та висновків до роботи.	26.05.2021 – 28.05.2021	Виконано
10	Оформлення дипломної роботи згідно вимог. Додання необхідних сторінок та подача роботи на рецензування.	31.05.2021 – 04.06.2021	Виконано

Студент Антон МАРІНОВ

Керівник Олександр РОМАНОВ

РЕФЕРАТ

Текстова частина дипломної роботи: 90 с., 53 рис., 1 табл., 13 джерел.

Актуальність. Для високої доступності сервісів, послуг використовується технологія кластеризації, що дозволяє розподіляти однакові задачі на декілька обчислювальних вузлів. Прикладами мереж для інноваційних технологій, функцій, являються мережі з підтримкою технології SDN (програмно-конфігурованих мереж). Дана технологія дозволяє спростити керування мережею, перенесенням управління мережі на єдиний мережевий пристрій – контролер, вміючи налаштовувати (програмувати) контролер, можливо досягти повної автономної роботи мережі. Також дана технологія підтримує у власних мережах використання технології віртуалізації, що дозволяє замінити громіздкі фізичні прилади віртуальними з такою ж або навіть більшою продуктивністю.

Мета. Ціллю даної роботи являється здобуття необхідних знань та навичок для побудови як найпростіших так і віртуальних мереж більш складних структур, топологій. Також ще однією важливою ціллю являється набуття розуміння, щодо роботи та застосування технологій віртуалізації і кластеризації у сучасних мережах, а також використання та керування мережею, використовуючи віддалені додатки, пристрої (контролери).

Методи. Наглядне створення та наведення прикладів запуску різноманітних мереж на базі платформи Mininet та можливості дослідження на даній платформі. Використання та підключення віддаленого віртуального контролеру для керування програмно-конфігурованими мережами. Створення цільного макету кластерної мережі ЦОД з використанням платформи Mininet та віддаленого контролеру HPE VAN SDN.

Результат. За допомогою програмного емулятору Mininet спроектовано кластерну мережу ЦОД у вигляді SDN мережі. За допомогою підключення програмного віддаленого контролеру HPE VAN SDN керування мережею перенесено на нього та з’явилась можливість переглянути стан мережі через

веб-інтерфейс. Визначені переваги та недоліки використання інтерфейсу Mininet та віртуального контролеру даної моделі.

Ключові слова: Software Defined Network**,** SDN, ЦОД, Mininet, OpenFlow, HPE VAN SDN Controller, Open Virtual Switch, віртуалізація, кластеризація.

ABSTRACT

Text part of diploma thesis consists of: 90 p., 53 fig., 1 table, 13 sources

Significance: For high availability of services use clustering technology, which allows to distribute the same tasks to multiple computing nodes. Examples of networks for innovative technologies, functions, are networks with support of SDN technology (software-defined networks). This technology simplifies network management by transferring network control to a single centralized network device - the controller, being able to configure (program) the controller, it is possible to achieve full autonomous network performance. This technology also supports the use of virtualization technology in its own networks, which allows to replace physical devices with virtual ones with the same or even higher performance.

Objective: The purpose of this work is to acquire the necessary knowledge and skills to build both the simplest and virtual networks of more complex structures, topologies. Another important goal is to gain an understanding of the operation and application of virtualization and clustering technologies in modern networks, as well as the use and management of the network using remote applications, devices (controllers).

Methods. Visually creating and providing examples of launching various networks based on the Mininet platform and research opportunities on this platform. Use and connect a remote virtual controller to manage software-defined networks. Creating a complete layout of a cluster data center network using the Mininet platform and the HPE VAN SDN remote controller.

Results. A cluster data center network in the form of SDN network was designed using the Mininet software emulator. With the connection of the HPE VAN SDN remote control software, network management was transferred to it and was possible to review the network status using the controller web interface. Were determined advantages and disadvantages of using the Mininet interface and the virtual controller of this model.

Keywords: Software Defined Network**,** SDN, DPC, Mininet, OpenFlow, HPE VAN SDN Controller, Open Virtual Switch, virtualization, clustering.

ЗМІСТ РОБОТИ

СПИСОК СКОРОЧЕНЬ

ЦОД – центр обробки даних;

SDN – Software Defined Network – програмно-конфігурована мережа;

OF – OpenFlow;

sw – switch – комутатор;

h – host – хост;

LAN – Local Area Network – локальна комп’ютерна мережа;

API – Application Programmable Interface – інтерфейс прикладного програмування;

VXLAN – Virtual Extensible Local Area Network;

ОС – операційна система.

ВСТУП

В наші часи інформація стає потужним інструментом для знань, роботи, розвитку, та навіть інколи в вигляді потужної зброї проти ворога. Адже інформація містить у собі різноманітні відомості про будь-що у світі, будь-то телекомунікації, медицина, історія. Інформація з кожним днем набуває все більших і більших обсягів і тим самим зберігати її стає важче. Слід враховувати, що велика кількість інформації є конфіденційною (секретною), та її слід оберігати необхідним чином. Деяку інформацію можливо отримати заплативши за неї, таку інформацію називають послугою (підписки на різноманітні сервіси, інтернет, телебачення).

Крім надання і засекреченням/захистом інформації слід також не забувати про її зберігання, резервування та забезпечення її максимальної доступності.

Щоб забезпечити цю саму конфіденційність, належне зберігання та доступність інформації використовуються найсучасніші технології, мережі. Однією з таких мереж, комплексів і являється центр обробки даних, який і допомагає забезпечувати стабільну роботи систем, що працюють над обробкою, наданням і захистом інформації, виділяючи усе необхідне обладнання для розгортання і роботи цих самих систем. Для стабільної роботи слід зі сторони замовника правильно налаштувати мережу за допомогою спеціалістів, або ж, якщо це доступно, з допомогою спеціалістів того самого центру обробки даних. Бачимо, що відповідальність за доступність інформації, якість її надання та безпеку, лягає на плечі не лише команду використовуваного центру даних, а й на сторону замовника. Оскільки аренда обладнання не вирішить всі проблеми та аспекти роботи мережі, варто знатися та правильно підійти до налаштування мережі (за допомогою спеціалістів), підтримки мережі, поступової модернізації, оновлення, розширення мережі. Але це вже більше маркетинг, продовження міркувань залишимо відповідним спеціалістам, а ми зупиняємося на технічній частині.

Оскільки в нас темою для розгляду являється кластерний центр обробки даних на базі технології програмно-конфігурованих мереж. Слід розбити усю тему на декілька частин та розібрати кожну з них окрему, аби мати розуміння роботи кожного компонента і усієї мережі в цілому. Що спростить сприймання практичного матеріалу, способу реалізації та підвищить розуміння та знання в майбутніх дискусіях на схожі теми.

РОЗДІЛ №1. Аналіз принципів побудови центрів обробки даних

1.1. Призначення центрів обробки даних

Спочатку перейдемо до розгляду мереж, навколо яких усе і розгортається. А саме мереж центру обробки даних (ЦОД).

Багато хто вважає, що ЦОД – це спеціалізоване приміщення, в якому компанія розміщує власне мережеве чи серверне обладнання для майбутнього підключення клієнтів до мережі Інтернет. І так і ні, адже це загальне розуміння даного терміну. Але ЦОД – це ціла система, яка складається з багатьох мережевих елементів, котрі тісно пов’язані між собою і працюють разом над виконанням спільної задачі.

Робота та мета ЦОД закладається в обслуговуванні клієнтів, а саме надання в володіння (оренду) провайдерам/компаніям власне мережеве (серверне) обладнання, на якому клієнт можу розгорнути будь-яку власну систему, мережу. Однак ЦОД можуть ще обслуговувати приватних клієнтів, наприклад виділяти клієнту повністю сервер, або місце під сервер клієнта (клієнт ставить власне обладнання), які в свою чергу мають власних користувачів та частіше за все надають їм ряд певних послуг. Також будь-який дата-центр при необхідному налаштуванні та підході підтримує та забезпечує захищені канали зв’язку, по яким і відбувається обмін даними (частіше за все конфіденційними).

ЦОД володіє насамперед функціями забезпечення стабільної і безвідмовної роботи обладнання, яке активно використовується іншими компаніями-споживачами або користувачами.

По сьогодення, мережі ЦОД демонструють стрімкий ріст в використанні та ці мережі перетворюються в ключовий елемент корпоративної інформаційно-обчислювальної інфраструктури. Особливо при обслуговуванні критично важливих сервісних систем і управлінні ключовими інформаційними ресурсами.

1.2. Структура системи ЦОД і призначення її основних елементів

Коли ми знаємо визначення та призначення систем ЦОД, треба перейти до розуміння практичної роботи даної системи. Для цього слід поглибитись та розглянути з чого складається уся мережа ЦОД та кожний елемент ЦОД і

які його обов’язки в системі. ЦОД може складатися з наступних мережевих елементів: ядра усієї мережі, серверів, дискових накопичувачів, комутаторів, маршрутизаторів, систем безперебійного живлення та керуючих комп’ютерів.

Вищезазначені елементи можна спостерігати у вигляді структури системи на рисунку нижче:

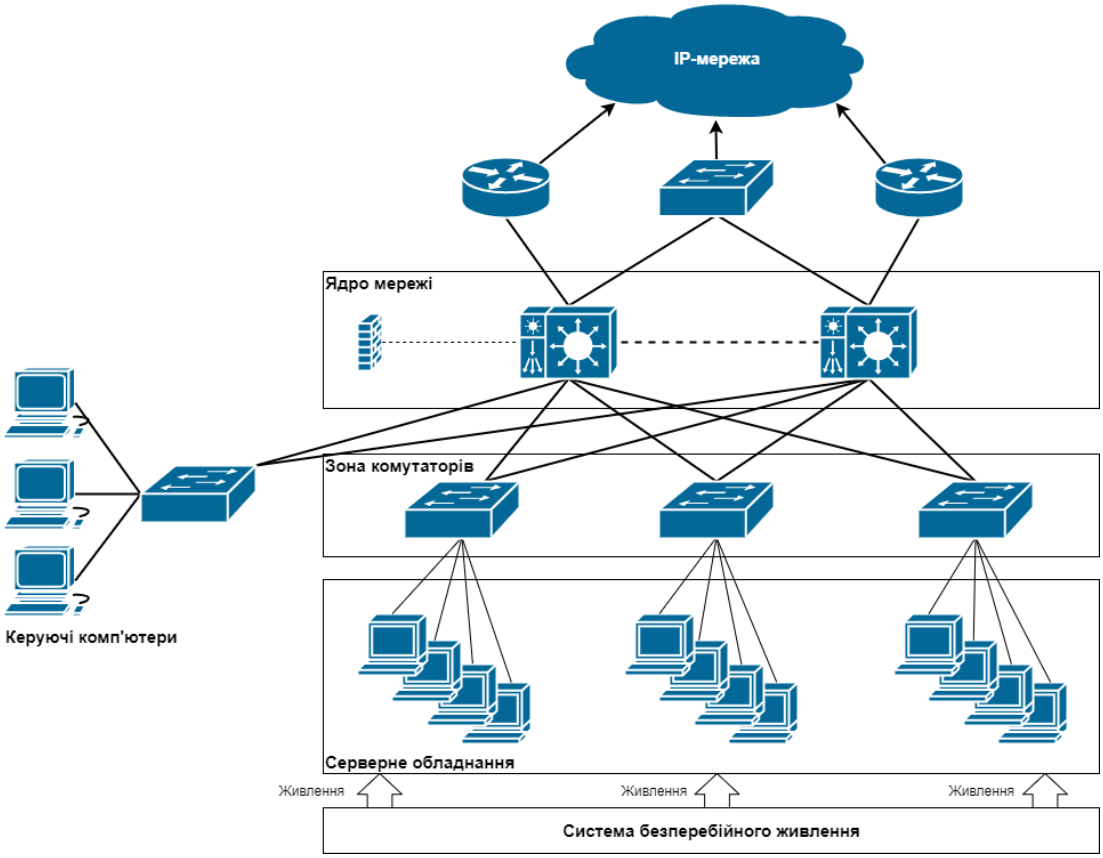


Рисунок 1.1 – Структурна схеми типової ЦОД мережі

Зображена вище структурна схема мережі, розподілена на зони, для більш зручного сприйняття та розгляду системних елементів.

Розглянемо роботу системи ЦОД більш детально. Вона полягає в тому, що серверне обладнання, виділяється відповідному замовнику (клієнту), який розгортає на цьому обладнанні власну мережу, або власні комп’ютери, віртуальні комп’ютери, сервери, для досягання власних цілей.

Комутатори в такій системі спрощують спільну роботу серверів, котрі виконують схожі задачі, але якщо навіть і сервер одинарний, він все одно під’єднаний до комутатора для належної роботи. Також комутатори

пересилають трафік до серверів (запити) та від серверів (відповіді, контент) на комутуючі модулі.

Комутуючі модулі, частіше за все, на чолі з контролером, оброблюють інформацію (трафік), визначаючи куди його надсилати (або на який комутатор, або на який маршрутизатор) і надсилають, відповідно рішенню (дані рішення, також називають “політиками”, фаєрволами (мережевими екранами), можуть генеруватися автоматичного відповідно роботі контролера, або ж створюватись вручну, на цьому самому контролері).

Маршрутизатор/и, згідно отримувача інформації в відомостях пакету, відсилає пакет/и до відкритої мережі “Інтернет” за певною логічною адресою, за якою знаходиться клієнт (замовник) або клієнт замовника.

Також не слід забувати про керуючі комп’ютери, за допомогою яких персонал центру даних керує усією системою з можливостями моніторингу, конфігурації та підтримки тієї чи іншої користувацької, або власної мережі.

Система безперебійного живлення безпосередньо критично важлива ділянка у системі, адже за допомогою її живляться усі мережеві компоненти і не підлягають знищенню при великих навантаженнях, які можуть спричинити вимкнення усієї системи.

Тепер розглянемо більш детально кожний мережевий елемент і дізнаємося як же в такій системі всі компоненти взаємодіють між собою.

Ну спочатку вмикається система безперебійного з’єднання, яка живить роботу усього ЦОД. Дана система просто живить мережу, тому ми її не будемо більш детально розглядати.

Серверне обладнання. Тут в нас розташовані сервери, які розділені по групам (кластерам), в залежності від обов’язків (працюють над вирішенням різних задач). Під обов’язками розуміється: робота з запитами, що поступають на мережу від клієнтів, виконання обчислювань і т.п. Окремо сервер складається з: фізичного сервера (коробка з апаратним обладнанням), віртуальних машин та віртуального комутатора, який безпосередньо поєднує віртуальні машини. Дані віртуальні машини і працюють над виконанням задач, на кожній віртуальній машині виконується певна задача. Кількість віртуальних машин залежить сугубо від апаратної конфігурації фізичного серверу.

Архітектура класичного сервера у системі центру обробки даних зображена нижче:

Рисунок 1.2 – Архітектура серверного вузла у ЦОД

У сервера, також є можливість під’єднатися одразу до 2-х найближчих комутаторів. Що збільшить безпеку та резервування інформації, адже можна наприклад активувати функцію екранування, що буде дублювати та шифрувати необхідний канал з трафіком.

Тепер розглянемо призначення та принципи роботи кожної окремої ділянки/зони у системі центру обробки даних.

Зона комутаторів. Тут комутатори об’єднують сервери до груп, відповідно задач, і дає їм змогу спілкуватися один з одним. Комутатори також передають трафік до ядра системи. Кожен з комутаторів має 2 канали з’єднання з ядром мережі, що підвищує надійність мережі.

Зліва на схемі можна спостерігати також окремий комутатор, який поєднує ЦОД мережу з комп’ютерами керування. За допомогою даних комп’ютерів конфігуруються, налаштовуються під замовників сервери, комутатори, а також ядро мережі адміністраторами, які працюють на даних центр даних. Або може виконуватись моніторинг роботи мережі, її окремих елементів, або відстеження якості трафіку.

Серверне обладнання (зона серверів). Тут сервери, частіше за все, виконують обчислення або виконання певних задач, або ж обробку інформації, яка поступає на них, відповідають на неї та надають послуги користувачам, які замовляють, споживають послуги у орендаторів серверів.

Ядро системи. В дану частину входять: модулі передачі контенту (Content Switching Module), різні функції мережі (мережеві екрани, балансування навантаження і т.п.). Модуль комутації контенту забезпечує

високоефективне балансування навантаження сервера між групами серверів, брандмауерами, та іншими мережевими пристроями, які працюють на рівні 3, а також від 4 до 7 рівня інформації про пакет (по моделі OSI). В склад такого модуль також входять контролери, які мають змогу налаштовувати роботу системи, вносити конфігураційні корективи (налаштування маршруту трафіку, налаштування комутаторів). Дані модулі під’єднані напряму до комутаторів.

У верхній частині архітектури, можемо спостерігати таку “перехідну” ділянку, яка містить в собі: маршрутизатори та комутатори, які під’єднані до ядра системи. Вони приймають/передають весь контент, трафік на обробку до серверів та доставляють оброблену інформацію (відповіді) на клієнта, відповідно, через логічну мережу. Це одна з найголовніших зон мережі, тому було прийнято рішення продублювати маршрутизатор для збільшення доступності і надійності системи. Тому і встановлено 2 потужних маршрутизатора, які працюють незалежно один від одного.

Обидва маршрутизатора з'єднані з комутаторами двох робочих зон, розташованих на деякій відстані один від одного. Оскільки кожен комутатор має два канали зв'язку з центральним вузлом комутації (ядро мережі), обрив однієї з ліній зв'язку між ними або відмова одного з центральних пристроїв мережі не є критичним для роботи системи.

Архітектури з резервуванням (додатковими/резервними з’єднаннями), підвищує вимоги до каналів зв’язку (фізичних) та стійкістю до перешкод.

Багато хто напевне уявляє, бо колись побачив/-ла на картинках, у фільмах, що ЦОД складається з великої кількості великих шаф (стійок). Майже усі мережеві елементи центру даних поміщуються в такі стійки. Згадаємо, що сервери поділяються на групи, які ще називають кластерами, в залежності від виконуваної задачі. З такої групи усі сервери з’єднуються з одним і тим же комутатором і поміщуються до однієї шафи, однак таких груп може бути декілька в одній стійці. Також сервери під’єднані відповідно до файлових накопичувачів, на яких сервери розгортають власні віртуальні машини, віртуальні комутатори та необхідні додатки. В такій стійці також розташована частина безперебійної системи живлення, яка підтримує роботу елементів в рамках однієї стійки. Та для безпеки до такої стійки нерідко залучають підключення системи пожежогасіння, аби при критичній аварії, можна було запобігти загоряння у шафі і якщо не цілком, то хоча б частково, зберегти дані мережевих компонентів.

Отже, побудована стійка ЦОД виглядатиме у виді наступної структури і включатиме в себе наступні архітектурні елементи мережі ЦОД:

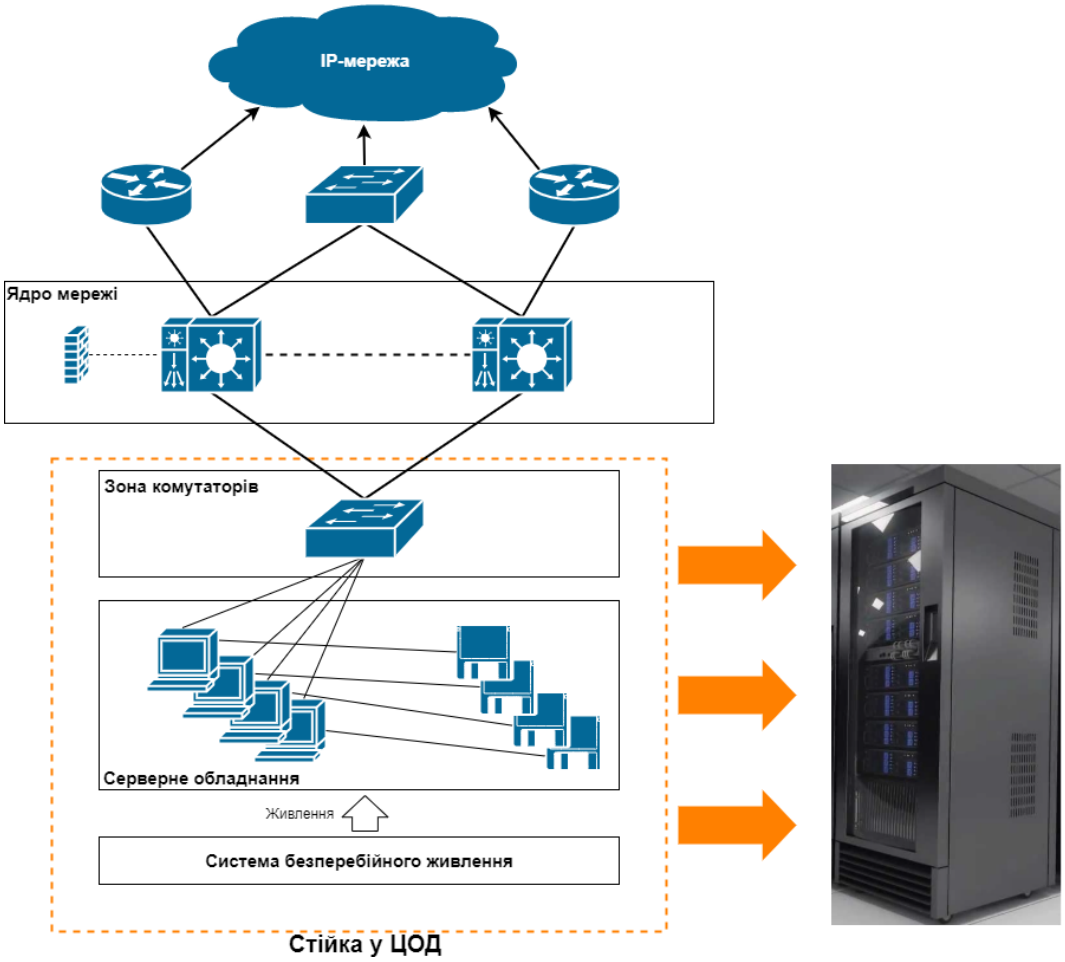


Рисунок 1.3 – Архітектура стійки (шафи) у системі ЦОД

В даній структурі можемо спостерігати, що використовуються зовнішні файлові накопичувачі. Кожен з яких під'єднаний до серверу. А сервер, в свою чергу, використовує даний файловий накопичувач для розгортання віртуальних машин та їх роботи над певними задачами.

Під час розширення корпоративної архітектури ЦОД, в дата центрах розгортаються різноманітні види нових додатків, завдяки з однієї сторони, прогресу в технологіях телекомунікаційних мереж, а з іншої сторони, зростаючої популярності інтернету і вимог користувачів.

Всі ці проривні зміни та прогреси не тільки підсилюють залежність компаній від ЦОД, але й прискорюються появлення стратегії реалізації ЦОД з масштабованими оновленнями.

1.3. Особливості побудови мережі ЦОД с використанням SDN

Для того, аби мережі ЦОД могли встигати за достатньо стрімким ростом даних, сервісів та новим рівнем ресурсів, вимагались нові можливості серверів як для зберігання, так і для обробки все більш і більш складних сервісних взаємодій. Однак, як і передбачалося, архітектури і технології традиційних ЦОД не відповідали новим вимогам щодо високої ефективності, інтелектуальності та зручності роботи. Цей стратегічний виклик і призвів до появи технології програмно-конфігурованих мереж (SDN).

В даній роботі, ціллю являється розглянути різноманітні технології в мережах, які спрощують керування мережею, підвищують безпеку та доступність мереж. Однією з таких технологій являється програмно-конфігуровані мережі, про яку буде розказано трохи пізніше. Та другою ціллю являється знайти способи перетворення певної класичної кластерної мережі ЦОД на віртуальну SDN ЦОД мережу.

Перед нами стоїть задача створити лабораторний макет на базі програмно-конфігурованих мереж, який наближений до реальної кластерної мережі центру обробки даних, та відображає наближену до реальної роботу мережі. Після чого будуть проведені необхідні дослідження, щодо параметрів мережі.

Для даного макету було уявлено мережу, макет якої виглядатиме наступним чином:

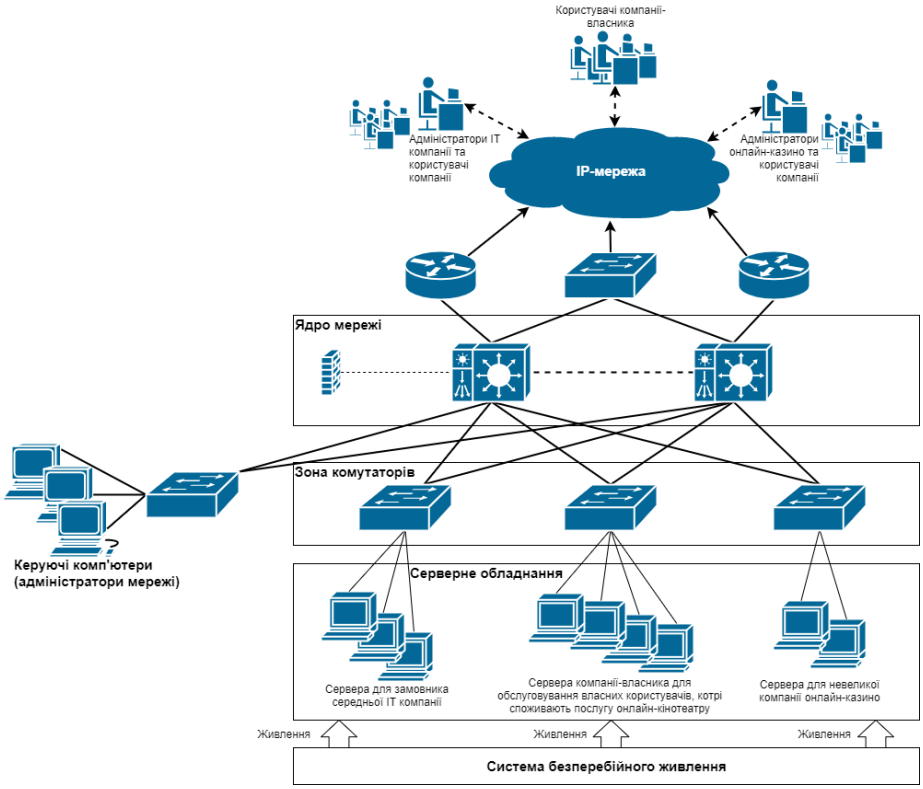


Рисунок 1.4 – Приклад мережі для подальшої реалізації на базі SDN

Суть полягає в тому, що існує та працює компанія-ЦОД, яка надає власні послуги (онлайн кінотеатру) і надає іншим компаніям послуги хостингу. У ЦОД з’явилося 2 компанії-замовника, котрі використовують сервіси хостингу. Робота компанії-ЦОД закладається у наданні та обслуговувані мереж та клієнтів компаній-замовників, і наданні контенту власного сервісу власним клієнтам. В якості серверів, користувачів та адміністраторів в мережі використовувались звичайні віртуальні хости емулятора Mininet. Ціль такої мережі – доступність отримання послуг клієнтами, після звернення (подання заявки) на сервер. Дану топологію мережі буде пізніше розділено для більш зручної та можливої побудови у вигляді віртуальної мережі на базі SDN.

1.4. Висновки до першого розділу

В даному розділі було визначено призначення ЦОД – системи, що складається з багатьох мережевих елементів, котрі тісно пов’язані між собою і працюють разом над виконанням, вирішенням спільних задач у мережі. Розглянуто структуру типової мережі центру обробки даних та призначення

її мережевих елементів. Було розглянуто типову “шафу”, яка присутня у багатьох дата-центрах і з яких елементів структури системи ЦОД вона складається. Було визначено задачу на практичну частину роботи.

РОЗДІЛ №2. Принцип роботи кластерної мережі. Технології SDN, віртуалізації та кластеризації у мережах. Шляхи побудови кластерної мережі центру обробки даних на базі SDN

2.1. Кластерна мережа. Принцип побудови кластерної мережі ЦОД

Як вже було вище зазначено, що групи об’єднаних серверів (або вузлів мережі) по якомусь принципу називають кластером. Тому доречно буде зазначити, що таке кластер і кластерна мережа.

Кластером в мережі називають – сукупність пов’язаних (з’єднаних) мережевих компонентів, котрі працюють над вирішенням спільної задачі, спільним обчисленням.

Кластерною мережею, відповідно, виступає сукупність кластерів в межах однієї мережі. Кластери у таких мережах отримують власні задачі і працюють над їх вирішенням. Додатково, за допомогою технології кластеризації, мережеві елементи в рамках одного кластеру отримують підвищену доступність і резервність, що дає змогу кластеру надавати безвідмовну роботу впродовж довгого часу.

У випадку ЦОД, можна сказати, що це дійсно кластерна мережа. Оскільки в традиційній ЦОД існує багато кластерів (груп серверів), які поєднуючись створюють цілісну кластерну мережу.

В кластері забезпечується велика доступність та надійність роботи мережі. Оскільки задачі елементів, що вийшли з ладу, розподіляються на інші вузли в кластері, тим самим користувачі не відучують різницю в обслуговуванні.

Під вузлами в кластері, або кластерній мережі, розуміють сервери, це якщо розглядати кластери у ЦОД. Взагалі вузлами в кластерній мережі можуть виступати: сервери, робочі станції, а також звичайні робочі персональні комп’ютери.

Під робочою станцію тут розуміється цілий комп’ютерний термінал (який включає в себе периферійні пристрої, які часто віддалені від керуючого комп’ютера), набір потрібного програмного забезпечення, також у нього

можуть бути присутні допоміжні обладнання. Даний термінал також часто може значитись у певній обчислювальній мережі, яка також може виступати у ролі кластеру.

На вузлах (хостах) частіше за все виконуються, вирішуються певні задачі (виконуються розрахунки, робота з документами, робота з базами даних). Якщо це ЦОД, то кластери в такій мережі виконують задачі, які замовив та вказав замовник, який арендує мережеве обладнання.

Візуально архітектура кластерної системи (мережі) може набуватиме наступного вигляду:

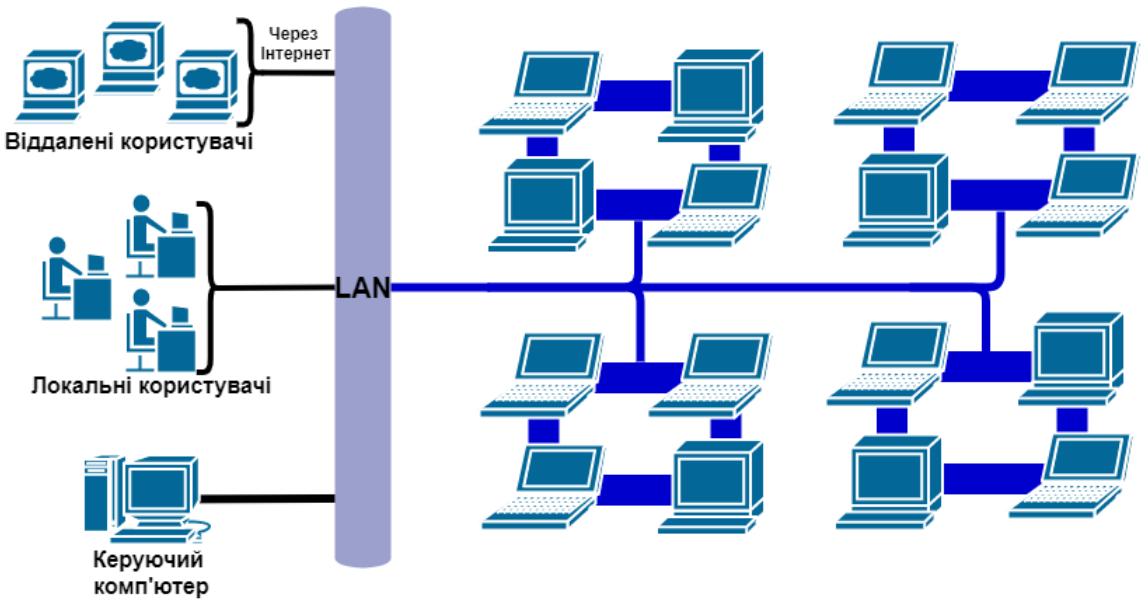


Рисунок 2.1 – Загальна архітектура кластерної мережі

У кластерній мережі застосовується технологія кластеризації. За допомогою даної технології і забезпечується можливість розподілення роботи між вузлами мережі для резервування даних.

Тобто, кластеризація – це роздача кластерам (одиницям кластерної мережі, вузлам) задач таким чином, щоб кожен кластер мав схожі задачі з іншими певними кластерами і в той же час мав відмінні задачі.

Розглянемо переваги кластерних мереж, а також технології кластеризації:

- можливості масштабованості кластерів дозволяють в багато разів збільшувати продуктивність задач, додатків в напрямку користувача на базі шинної архітектури або комутатора;

- кластеризація в загалом підвищує працездібність мережі в разі відмови якогось вузла в мережі, оскільки в такому випадку інший вузол бере на себе роботу/навантаження вимкненого пристрою. Тим самим це підтримує стабільність роботи для користувачів, які навіть і не помітять неполадки.

Існує також формальне визначення кластеризації, яке описане наступним чином: існують змінні X – множина задач, Y – множина компонентів.

Задано функцію відстані між задачами $\rho(x, x')$. Існує кінцева вибірка задач: $X^m = \{x_1, \ldots, x_m\} \in X$. Далі слід дану вибірку розбити на підмножини, котрі не пересікаються між собою, так, щоб кожний кластер складався з задач, які близькі по метриці ρ , а задачі кластерів щоб відрізнялися. Далі кожній задачі приписується номер кластеру y_i .

Тепер як працює алгоритм кластеризації. Опираючись на наші змінні, створюється функція $a: X \to Y$, яка кожній задачі $x \in X$ ставить у відповідність номер кластеру $y \in Y$.

Це все можна більш наглядно пояснити за допомогою наступного зображення:

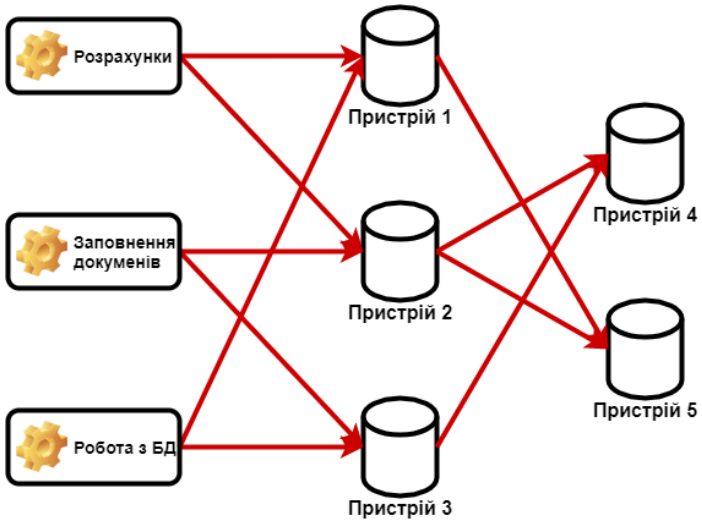


Рисунок 2.2 – Технологія кластеризації
(на прикладі кластерної комп’ютерної мережі)

На рисунку вище ми можемо спостерігати, що існує кластерна мережа, яка виконує певні задачі: розрахунки параметрів, заповнення документації та роботу з базою даних. Ці задачі рівномірно розподілені між пристроями

таким чином, щоб у разі виходу з ладу одного пристрою, інші пристрої рівномірно розподіляли між собою його обов’язки і мережа продовжувала функціонувати в звичному режимі. Слід помітити, що усі 3 задачі не призначаються одному пристрою, оскільки це максимально навантажить вузол і в разі виходу з ладу іншого вузла, він не зможе взяти на себе його обов’язки, оскільки в такому випадку він перевантажиться.

Якщо влаштувати реальну ситуацію, а саме відмову одного пристрою. Та дії, які були описані вище, то візуально це можна зобразити наступним чином:

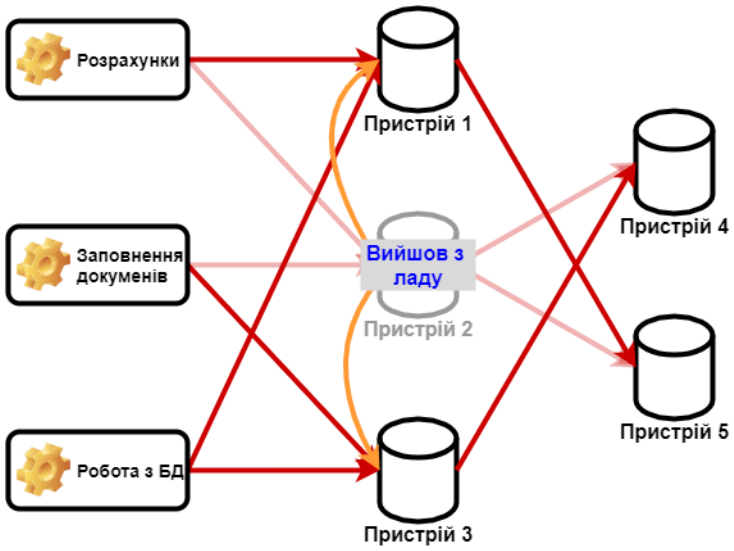


Рисунок 2.3 – Робота мережі при відмові пристрою

Кластеризація може виконуватись на багатьох рівнях в комп’ютерній системі. Це можуть бути: операційна система, системи керування, апаратне забезпечення, програми(додатки). Чим більше рівнів охоплює кластерна технологія – тим вище надійність, масштабованість та гнучкість керування (але в той же час і зростає навантаження на мережеві компоненти).

Тобто, побудова кластерної мережі ЦОД означає – побудову такої мережі, в якій кластери мережевих вузлів забезпечували б високу доступність та продуктивність роботи мережі на усіх ділянках, навіть при виході з ладу певних мережевих компонентів. Задачі яких миттєво розподіляються на вузли в певному кластері і підтримують стабільну роботу мережі, використовуючи більше обчислювальної потужності.

2.2. Побудова кластерної мережі ЦОД з використанням технології SDN та розгляд її мережевих компонентів

Як вже було трохи вище зазначено, що мережі дата центрів не завжди встигали за стрімким розвитком даних, інформації, методи її обробки, зберігання. Для усіх цих так званих “новинок” необхідно було забезпечувати вкрай високу ефективність роботи мережі. Але згодом, з розвитком програмно-конфігурованих мереж (SDN-технології), ця проблема була вирішена у ЦОД і з того часу більшість дата центрів, по сьогодення використовують віртуальні мережі, мережеві елементи, віртуальні рішення для відповідності трендам і цим самим підкорюючи все більше і більше компаній у використанні таких ЦОД та необхідності, прибутковості використання віртуального обладнання або ж віртуальних ресурсів.

Тепер поговоримо про технологію SDN (програмно-конфігурованих мереж). Дана технологія, SDN мереж, забезпечує можливості мережевої віртуалізації. Вона розділила рівні керування та переадресації трафіку, реалізувала логічно централізоване управління (керується все через один пристрій за протоколом OpenFlow) і відкрила безліч можливостей мережі для додатків вищого рівня.

Головні цілі SDN – створити відкриту, програмовану та завжди придатну для змін віртуальну (або гібридну) мережу. Керування якою здійснюється віддалено. Також головною ціллю таких мереж стала можливість встановлювати різноманітні утиліти/функції для здійснення: управління, моніторингу трафіку, керування пріоритетами доставки, налаштування балансування навантаження, мережевих екранів (firewalls).

Слід в декількох словах пояснити що таке віртуалізація, адже технологія SDN, майже вся пов’язана на віртуалізації.

Спочатку дамо визначення терміну віртуалізації в комп’ютерних системах. Віртуалізацією являється виділення обчислювальних ресурсів комп’ютера (хоста) для віртуальних задач, незалежність їх від апаратної частини комп’ютера та ізольованість обчислювальних задач, процесів між собою, які виконуються на фізичному пристрої.

Віртуалізація дуже поширена та розвинена в наші часи. Її все частіше і частіше використовуються для заміни громіздких фізичних пристроїв в мережі на більш перспективні віртуальні, для паралельного запуску декількох операційних систем на вузлі чи сервері, для створення та зберігання ресурсів на віртуальних сховищах, для побудови та запуску віртуальних мереж.

Тобто віртуалізація це процес створення чогось “штучного”, що з використанням реальних обчислювальних ресурсів стає реальним і часто являється кращим, перспективнішим аналогом фізичного джерела.

Візуальне зображення віртуалізації компонентів на гіпервізорі зображено на рисунку нижче:

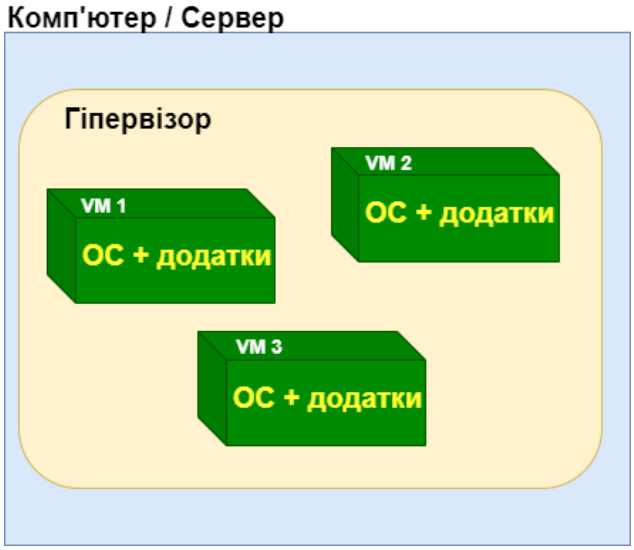


Рисунок 2.4 – Приклад технології віртуалізації на базі гіпервізору

Існує багато типів віртуалізації, а саме:

Обладнання. Це так звана емуляція, або ж повна копія реального приладу, або цілої платформи. Тут одразу можна в прикладі привести OVS (віртуальний комутатор), який являється повною копією фізичного комутатора. Але при підтримці технологій, він має значно більше можливостей та функцій в порівнянні з оригінальним фізичним комутатором.

Програмне забезпечення. Сюди входять віртуалізація додатків – робота додатків в середовищі, що незалежне від операційної системи, та віртуалізація сервісів – робота системних компонентів, які необхідні при запуску додатків, функцій в системі, в даному випадку віртуалізація відбирає лише найнеобхідніші елементи в таких сервісах.

Операційні системи. Тут також декілька підвидів віртуалізації:
Програмна віртуалізація:

- Тут може відбуватися динамічна трансляція, це коли якісь проблемні елементи, або процеси гостьової операційної системи підтримуються гіпервізором.

- А також паравіртуалізація, це коли основна операційна система взаємодіє з програмою-гіпервізором, який в свою чергу ділиться гостьовим API, замість використання ресурсів напряму (тобто ресурси використовуються по мірі необхідності, це контролюється через API).

Апаратна віртуалізація. Це віртуалізація зі спеціальною підтримкою процесорної архітектури. В даній віртуалізації гостьова операційна система не залежить від основної системи, що дозволяє забезпечити продуктивність реального комп'ютера, що дає використовувати машину для повноцінних задач. Найбільш розповсюдженими технологіями апаратної віртуалізації являються Intel-VT та AMD-T, тобто технології, які розробляються виробниками процесорів Intel та AMD, відповідно.

Віртуалізація на операційному рівні. Дана віртуалізація означає паралельну роботу декількох користувацьких просторів на одній операційні системі. Прикладом може виступати паралельна робота програмних продуктів Docker і LXC на операційній системі Windows 10. В даному випадку створюються окремі оболонки (простори) для Docker (програмне забезпечення для роботи з додатками з використанням контейнерів) та LXC (Linux Containers – система для запуску елементів операційної системи Linux на одному пристрої та взаємодія з ними). Тобто ці програми не заважають один одній і в той же час можуть паралельно використовуватись користувачем Windows для вирішення певних задач.

Системи зберігання (простори зберігання).

Віртуальна файлова система (VFS) – система, що надає доступ клієнтським додаткам до відповідних різноманітних файлових систем.

Розподільна файлова система – система, яка надає можливість з різних пристроїв отримувати доступ до файлів в системі за допомогою підключення через мережу.

Гіпервізор зберігання – програма, котра керує віртуальним середовищем для зберігання даних. Також може об'єднувати різні фізичні простори зберігання у єдину систему.

Віртуалізація пристроїв зберігання даних. Це технологія віртуалізації жорсткого чи оптичного диску.

Мережі. Сюди входить віртуалізація мережі, яка може бути внутрішньою або зовнішньою. Під віртуалізацією мережі розуміється об'єднання, поєднання апаратних та програмних ресурсів, компонентів до єдиної цільної віртуальної мережі. Зовнішня мережа об'єднує багато мереж у одну віртуальну. Внутрішня, в свою чергу, створює віртуальну мережу у

рамках однієї системи (відповідно використовує елементи та ресурси саме цієї системи).

Також існує можливість створення віртуальних приватних мереж (VPN), що являє собою створення мережевих шляхів (з'єднань) поверх будь-якої іншої мережі, що дозволяє захистити власні дані користуючись інтернетом. Прикладом таких приватних мереж виступають так звані vpn-розширення (extentions) у веб-браузерах, які дозволяють поверх нашого підключення з провайдером створити інше мережеве підключення, видавши нашому підключенню до інтернет логічну адресу іншої країни. Тим самим надаючи змогу використовувати сервіси, які по якимось причинам недоступні в певних країнах. Або такі мережі добре слугують для отримання доступу до файлів, документів, сервісів вашої організації, звичайно використовуючи шифрування та захищене підключення. Яке частіше за все керується адміністраторами організації.

Приклад: у мене на роботі комп'ютер облаштований vpn-додатком від мережевого вендора Cisco. Вмикаючи який, лише пройшовши етап ініціалізування особи (ввівши власні дані), створюється мережа. Вона не видає логічну адресу іншої країни, ми можемо спокійно користуватися мережею інтернет. Але з підключенням до такої приватної мережі у нас з'являється доступ до доменів, мережевих адрес, на яких зберігається, оновлюється та оброблюється важлива, необхідна для роботи інформація та працюють необхідні для роботи сервіси.

Для порівняння традиційної ЦОД мережі і ЦОД на базі SDN можна згадати сувору реальність, а саме, що налаштування та ще й підтримка середнього ЦОД, як мінімум, вимагає багатьох зусиль, знань та коштів. Адже в дану послугу входять також: налаштування роутерів, під'єднання функцій: балансування навантаження (трафіку), захист мережі і т.п. Не слід забувати про те, що кожен пристрій в мережі налаштовується окремо.

Також багато компанії, де компанії типу ЦОД не виняток, намагаються зекономити, взяти обладнання десь дешевше, десь яке вже раніше використовувалось. І тут проявляється інша проблема. Інтерфейси налаштування мережевих пристроїв суттєво відрізняються, в залежності від виробника обладнання, серії, та іноді навіть моделі. Тобто, якщо ЦОД містить у власному арсеналі мережеві обладнання 2 або 3 різних виробників – для налаштування і підтримки усіх цих пристроїв необхідний спеціаліст, який знає як працювати з кожним інтерфейсом усіх виробників. Таких спеціалістів не так і багато на ринку, та й послуги їх коштують, відповідно, в рази більше від спеціаліста, який знається в роботі з обладнанням одного виробника.

Як вже було раніше зазначено, дана технологія розділяє один від одного керування мережею та передачу трафіку. Дані рівні називаються: control plane (керування передачею) і data plane (передача трафіку), відповідно. Рівень control plane відповідає за визначення, по яким правилам в мережі буде передаватися трафік. А вже на рівні data plane реалізовується передача трафіку, відповідно налаштованим правилам (політикам) на рівні керування.

Також, ще було зазначено, що все керування, в мережах даного типу, реалізується централізовано. Це означає, що рівень керування передачею трафіку (control plane) містить в собі централізований пристрій, який повністю автоматизує контроль мережі. Цим пристроєм в SDN виступає контролер. Він має окремий канал зв'язку з усіма пристроями в мережі, котрі підтримують протокол взаємодії OpenFlow, через які він дізнається найактуальніші відомості про мережу, її топологію, а також автоматично, по замовчуванню, складає таблиці маршрутизації та вивантажує їх на відповідні мережеві пристрої.

Візуальне розподілення рівнів у мережах з підтримкою технології SDN зображено на рисунку нижче:

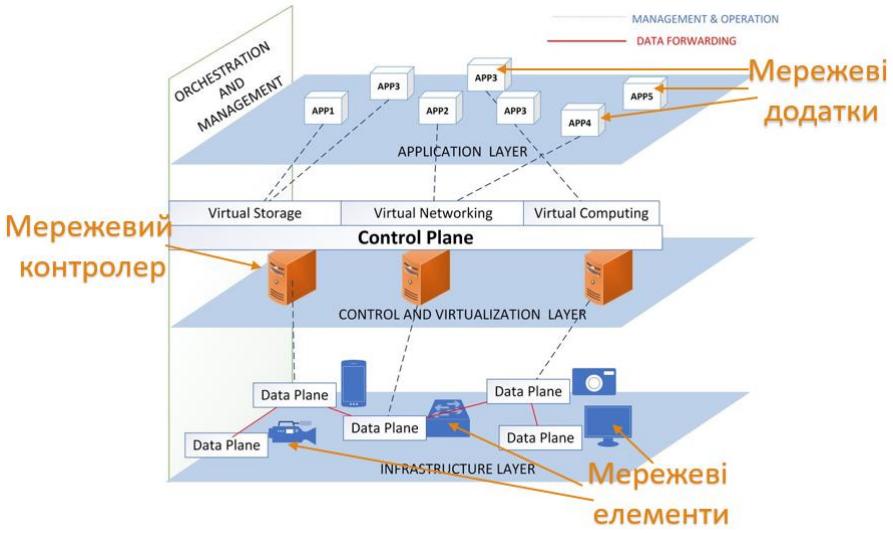


Рисунок 2.5 – Зображення основних компонентів SDN та розподілення їх між відповідними рівнями

Доречно зазначити, що весь зв'язок (між комутаторами та контролером) у мережі та усе керування мережею відбувається через протокол OpenFlow. Тому дамо визначення цьому протоколу.

OpenFlow протокол – протокол, який використовується для налаштування та адміністрування мережевих компонентів у програмно-конфігурованій мережі. За допомогою цього протоколу OpenFlow комутатори відправляють на контролер мережеву статистику. А комутатор, в свою чергу, завантажує за допомогою даного протоколу політики (правила) маршрутизації до кожного комутатору.

Протокол OpenFlow, в свою чергу, підтримує 3 типи повідомлень:

- контролер-комутатор – конфігурація і моніторинг стану комутаторів;
- симетричні повідомлення – такі повідомлення можуть відправляти як комутатор, так і контролер;
- асиметричні повідомлення – такі повідомлення відправляє комутатор контролеру, для повідомлення зміни власного стану.

Звісно можна і власноруч у мережеві пристрої вписувати нові правила, оновлювати таблиці переадресації. Це не так зручно як через контролер, оскільки забирає багато часу, але якщо контролер прописує правила не так, як хотілося, і немає можливості виправити контролер, оскільки немає знань з програмування контролеру, то ручне налаштування тут стає в нагоді.

Мережеві елементи SDN

Тепер розглянемо які ключові елементи присутні майже в кожній програмно конфігурованій мережі. Та розглянемо кожний елемент як незалежний пристрій.

SDN мережею прийнято вважати мережу, яка має в своєму складі: комутатор з підтримкою OpenFlow, хости (комп’ютери, сервери) та централізований SDN контролер.

Як вся ця мережа працює та як в ній взаємодіють мережеві елементи? В даній мережі хости підключаються через фізичні або віртуальні порти до OpenFlow комутаторів, які всі, в свою чергу, підключені та спілкуються (за допомогою OpenFlow протоколу) через окремий канал з контролером. Контролер під час спілкування з комутаторами збирає загальну інформацію про мережу, дізнається де розташовані усі мережеві пристрої. Після чого, контролер за допомогою додатків, які підключені до нього, розраховує найоптимальніші шляхи передачі і через спеціальні канали завантажує політики маршрутизації до таблиць потоків у кожний комутатор. Коли передається пакет, він потрапляє на комутатор, де в свою чергу потрапляє до таблиці потоків, під час проходження якої вирішується майбутній шлях пакету по мережі. І так відбувається на кожному комутаторів, поки пакет не дістанеться до отримувача.

Тепер більш детально подивимося на елементи мережі. Спочатку в нас по черзі йду мережевий елемент – *комутатор*.

Комутатори в даних мережах (програмно-конфігурованих) виступають в ролі пристроїв, які містять в собі таблицю, або декілька таблиць переадресації. Дані в таблиці регулярно оновлюються через контролер.

Архітектуру OpenFlow комутатора можна зобразити наступним чином:

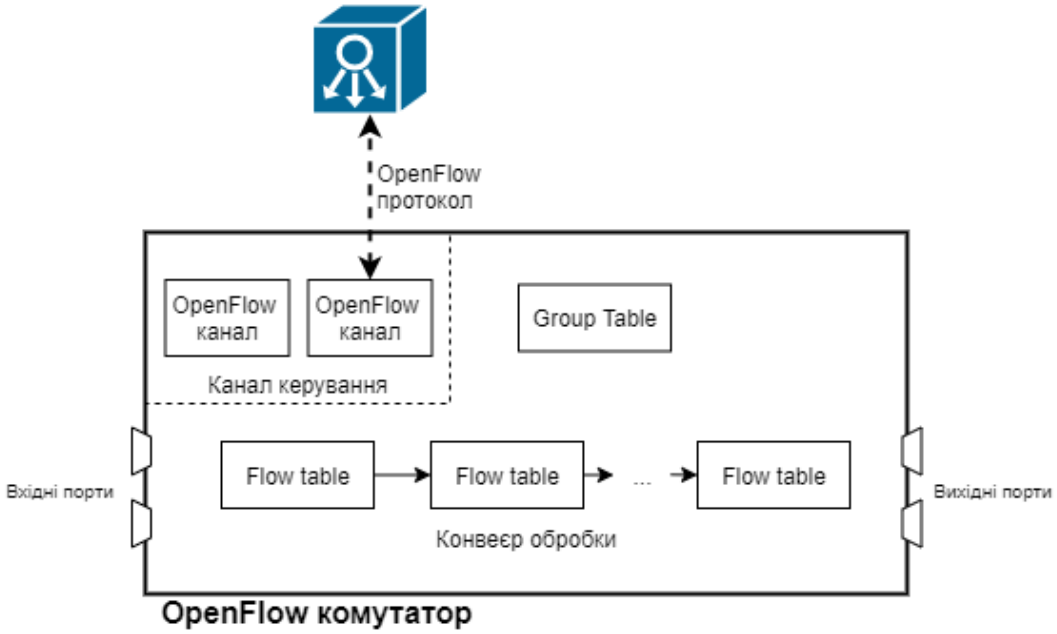


Рисунок 2.6 – Структурна схема комутатора з підтримкою OF протоколу

Бачимо на рисунку вище, що комутатор з підтримкою OpenFlow протоколу містить в своєму складі:

- канал керування. Тут через OpenFlow канал комутатор спілкується за допомогою OpenFlow протоколу з контролером (передає інформацію про стан мережі та отримує правила для роботи);
- конвеєр з таблиць потоків;
- групову таблицю.

Тепер розглянемо більш детально таблиці переадресації/потоків (flow tables) у програмно-конфігурованих комутаторах. Кожна таблиця містить в собі набір певних записів, кожний запис складається безпосередньо з: полів порівняння, лічильників та інструкції (дії, які необхідно застосувати до пакету).

Якщо декілька таблиць потоків в комутаторі – це називається конвеєром обробки.

Тепер розглянемо послідовність дій при надходженні пакету на вхідний порт комутатора. Візуально процес проходження пакета крізь комутатор можна побачити нижче на схемі.

Звірка в конвеєрі починається з нульової (по нумерації) таблиці. Звірка пакетів же виконується відповідно до пріоритету пакета. Якщо в таблиці адресації знайдено пакет (по полю порівняння) то застосовується до пакету відповідна інструкція. В якості інструкцій можуть бути дії, які описують переадресацію, модифікацію пакету, обробку групової таблиці та роботу конвеєра обробки. Сюди входять наступні дії з пакетом:

- Відправити пакет на відповідний вихідний порт;
- Відкинути пакет (drop);
- Відправити пакет до певної іншої таблиці потоків (goto);
- Змінити параметри пакету (логічну, фізичну адресу).

Перед застосуванням відповідної інструкції, поле лічильників оновлюється для необхідної статистики.

Якщо пакет не знайдено в таблиці, то подальші дії будуть залежати виключно від конфігурації комутатора. Тут можуть бути налаштування, аби пакет, який не знайшов себе у таблиці маршрутизації, був: відкинутий (dropped), переадресований до наступної (по номеру) таблиці для звірки (goto), відправлений до контролера, аби контролер прийняв рішення згідно пакету і записав інструкції щодо пакету до таблиці, для подальшої роботи.

Конвеєр обробки припиняє свою роботу тільки тоді, коли набір інструкцій, відповідний до запису таблиці, не вказує на передачу пакету до іншої таблиці (goto).

Тепер розглянемо групову таблицю. Це спеціальна таблиця, яка може бути, а може і не бути в комутаторі. Записи в такій таблиці називаються груповими правилами. Вони призначені для обробки пакетів з груповою адресацією чи широкомовною розсилкою. Запис в груповій таблиці містить правило, яке дублює пакети з певним заголовком, який відповідає шаблону і приписує дію над кожною копією такого пакету. Це все зроблено для того, аби знизити кількість правил в таблиці.

Послідовність обробки пакетів на OpenFlow комутаторі можна спостерігати на схемі нижче:

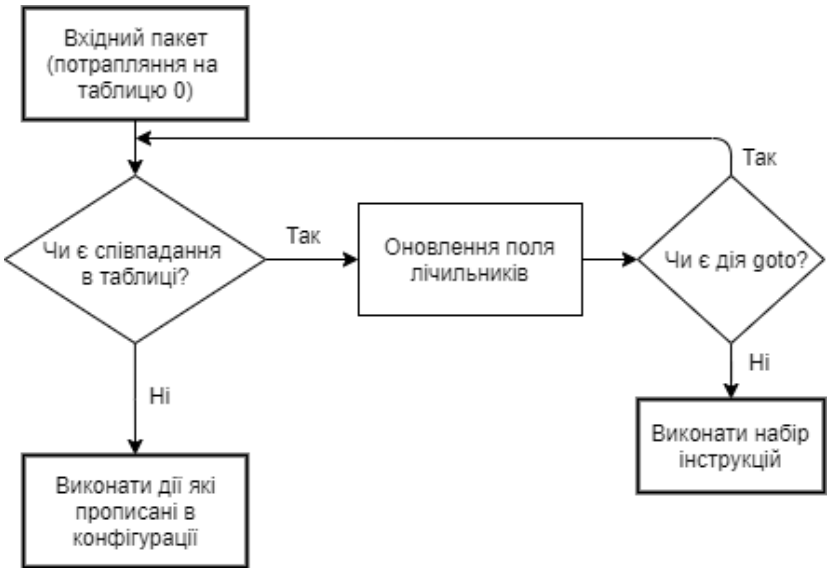


Рисунок 2.7 – Проходження пакету через ряд таблиць (конвеєр) в комутаторі OpenFlow

Комутатори в програмно-конфігурованих мережах можуть бути 2 типів: фізичні комутатори з підтримкою OpenFlow (так звані гібридні комутатори) та OpenFlow комутатори (OpenFlow only). Тобто гібридні комутатори можуть виконувати передачу звичного всім L2, L3 трафіку, так і бути частиною SDN мережі та повноцінно в ній взаємодіяти.

Чесно кажучи, під час досліджень та пошуку інформації, не траплялися на очі комутатори, котрі б не являлися гібридними. Найпопулярнішим гібридним віртуальним комутатором є *OVS – open virtual switch*. Він набагато складніший, хоча б за самою будовою, в порівнянні з тим самим фізичним комутатором, адже це багаторівневий комутатор. Розділяють такий комутатор на 2 основні частини: користувацький простір (user space) та простір ядра (kernel space).

Користувацький простір складається з: образів (daemons), які пов'язують сам комутатор з таблицею маршрутизаторів, ядро OVS та набору різних додатків, які допомагають налаштовувати комутатор, як завгодно, налаштувати його базу даних і напряму спілкуватися з ядром.

Модуль ядра в такому комутаторі запускає одразу 3 образи:

- Daemon (ovs-vswitchd) який відповідає за зміну і збереження конфігурації комутатора, також він зберігає зміни власної конфігурації в базу даних комутатора (Open vSwitch database - ovsdb). Даний образ і отримує свою конфігурацію з бази даних під час запуску. Він визначає

шляхи даних Open vSwitch, а потім здійснює перемикання через кожен міст, описаний у його конфігураційних файлах.;

- Database-server (ovsdb-server) який відповідає за керування базою даних та наборами потоків (на обробку). Цей образ забезпечує інтерфейси для однієї або декількох баз даних Open vSwitch (OVSDB). Він підтримує підключення клієнта через активні або пасивні сокети доменів TCP / IP або Unix;
- Bridge compatibility front-end for ovs-vswitchd (ovs-brcompatd) який відповідає за створення зв'язку між образами (daemons) та мостами системи Linux. Він це робить за допомогою прослуховування команд мосту і потім виконує додавання або видалення шляхів даних та інтерфейсів, які до них приєднуються.

До речі, образ ovs-vswitchd напряму під'єднаний до модуля ядра в обох напрямках (дуплекс) через протокол netlink. Це зроблено для того, аби при зміні конфігурації, зміни одразу були занесені до бази даних.

Більш того, даний комутатор дає можливість налаштувати різні функції, такі як:

- QOS (Quality of Service), наприклад коли треба обмежити пропускну здібність на певному напрямку;
- Mirroring, коли слід виконати моніторинг для безпеки. В такому випадку створюється дзеркальний порт (mirror port) і вибирається порт, пройдені через якого пакети будуть дублюватися на дзеркальний порт;
- Tunneling з використанням протоколу GRE (Generic Routing Encapsulation), який підтримує інкапсуляцію пакетів в так званому "тунелі". Тунель являє собою пряме з'єднання між двома вузлами мережі, застосовуючи інкапсуляцію протоколів. Інкапсуляцією тут виступає – процес включення всього пакету одного протоколу (тобто його заголовки і дані) всередину пакету іншого протоколу в якості переданої інформації.

Далі у черзі йде мережевий елемент – *контролер*.

Контролер розташований на рівні керування передачі (control plane), він завжди має найактуальнішу інформацію про стан мережі (працездібність пристроїв, топологію мережі, таблиці потоків).

На базі наявної інформації, додатки при контролері будують, необхідні для передачі, таблиці маршрутизації. Взаємодія контролеру з додатками реалізується через "Північний" інтерфейс, також можна підключати різні додатки, для певних цілей, це також все реалізується через даний інтерфейс.

Після побудови мережевих політик (таблиць маршрутизації) для комутаторів, їх слід завантажити на самі комутатори. Це робить контролер через “Південний” інтерфейс за допомогою OpenFlow комутатору. Тобто в SDN архітектурі присутній інтерфейс, який поєднує площини контролю і передачі даних. Взаємодія інтерфейсів між площинами зображена на рис. 2.8.

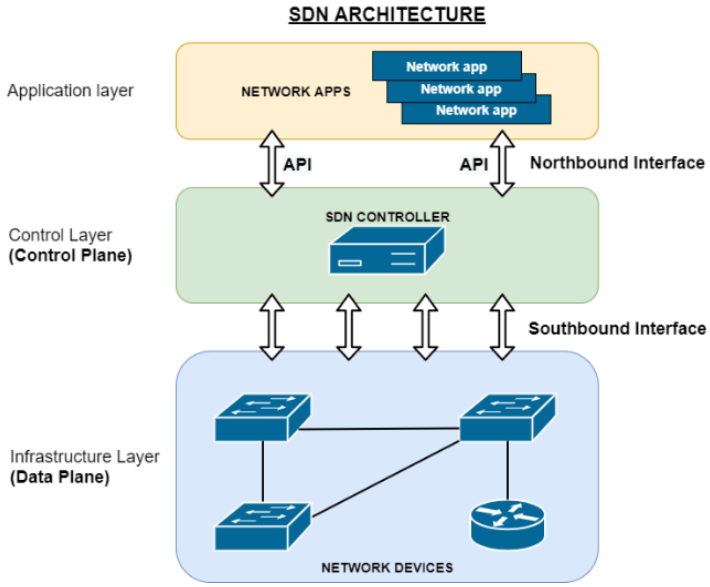


Рисунок 2.8 – Архітектура типової SDN мережі

Зараз існує безліч різноманітних контролерів, від різноманітних вендорів, багато контролерів розроблюються спеціально під певні вимоги замовника. Але частіше за все всі говорять та чули хоча би щось про POX, Open Daylight та Floodlight контролери. Ці три контролери і одночасно схожі, по порівнянням нижче, і в той же час такі різні. Порівняння їх можна спостерігати в табл.1.

Таблиця 1

Порівняння зазначених нижче SDN контролерів

	POX	Open Daylight	Floodlight
Підтримка мови	Python	Java	Java
Версія OpenFlow	v.1.0	v.1.0	v.1.0
Чи є OpenSource	Так	Так	Так

Чи є REST API	Ні	Так	Так
Підтримка платформи	Linux Windows Mac OS	Linux Windows Mac OS	Linux
Чи є GUI	Так	Так	Так
Підтримка GRE	Ні	Так	Так

Контролер POX.

Це продукт, який знаходиться у вільному доступі і розробляється під ліцензією GNU License на мові Python. Присутня підтримка OpenFlow версії 1.0, також присутня підтримка OVS extensions (розширення протоколу OpenFlow, що дозволяє робити більш тонкі налаштування комутаторів).

Сам контролер не керує трафіком, його основна ціль – запуск окремих компонентів, які виконують певний функціонал. Ці компоненти можна написати під свої вимоги на мові Python. Тобто даний контролер широко використовується розробниками.

Тобто це контролер, який використовується в більшості випадків для дослідження SDN і вирішує невеликі задачі. Повноцінним контролером, який працює в мережі, він не являється, оскільки достатньо обмежені можливості і повільна робота.

Контролер Open Daylight.

Це продукт, котрий має достатньо гнучку платформу, для розгортання плагінів, розробляться під ліцензією Eclipse. Розробляється на мові Java. В більш нових версіях контролер має підтримку OpenFlow v.1.3.

Даний контролер має можливість встановлювати, видаляти, змінювати плагіни без виключення самого контролеру. Він підтримує багату кількість мережевих протоколів. За допомогою даного контролеру стало можливо керувати паралельно різними областями мережі, використовуючи різні компоненти контролера. Також даний контролер має у власному арсеналі High Availability Model кластер, який дозволяє встановити контролер на декількох фізичних серверах, що дозволяє гарантувати відмовостійкість та балансування навантаження.

Отже, це контролер, що стрімко розвивається, як проект, який фінансує багато престижних компаній, що розробляють обладнання в сфері телекомунікацій. Оскільки контролер написаний на мові Java, то він являється кросплатформеним, тобто може використовуватись всюди, де є підтримка Java. Модульність надає можливість контролеру працювати

паралельно над декількома задачами, ділянками мережі, а також є можливість налаштовувати модулі контролера прямо від час його роботи.

Контролер Floodlight.

Це продукт, який знаходиться у вільному доступі і розробляється під ліцензією Apache на мові Java. Контролер набуває модульну архітектуру, це означає що перед його запуском, в конфігураційний файл контролера вписуються сервіси, функції, які будуть активовані при першому ж запуску контролера.

Цей контролер не підтримує циклічні топології (коли мережеві елементи замикаються і створюють петлю). Контролер підтримує безліч функцій, основні з яких:

- Forwarding – стандартне конфігурування OpenFlow комутаторів;
- Firewall – додання правил для більш гнучкої переадресації трафіку;
- Learning Switch – включає функції традиційного комутатора L2;
- Load Balancer – балансування різного трафіку (tcp, udp і т.п.).

Даний контролер вміє самостійно аналізувати мережу, відстежувати нові елементи в мережі. Також має в наявності свій зручний веб інтерфейс, на якому можна переглянути найголовніші параметри мережі.

Якщо підвести підсумки, то можна сказати, що порівняні контролери використовуються виключно для необхідних цілей. Якісь більше для дослідження, а якісь на більш практичному, робочому рівні.

Наприклад POX контролер може застосовуватись для тестування продуктивності програмно-конфігурованих мереж різноманітних топологій і тестування власних написаних компонентів. Але поступається через нестачу необхідної швидкості роботи і вимагання власного написання частини його інтелекту.

Open Daylight найсерйозніший проект/комутатор, оскільки на ньому можна запустити різноманітні топології мережі, існує підтримка багатьох протоколів мережевої передачі. Також, маючи модульність, контролер може виконувати безліч задач в одну і ту ж секунду. Ще можливо виконати конфігурування модулів, не вимикаючи, не зупиняючи роботу контролера. Саме цей контролер ідеально підходить для більш глибоких досліджень та непогано себе проявить при рішенні корпоративних, досить реальних задач. Даний контролер буде і надалі активно розвиватися та підтримуватись, оскільки цей проект наразі дуже перспективний, і має в спонсорах багато компаній зі спільноти Linux (наприклад: Cisco, ONF, IBM).

Floodlight контролер знаходиться посередині між вищезгаданими контролерами. Оскільки він також більш академічний, використовується активно в дослідженнях. Але має обмеження в плані побудови мережі, оскільки топології з петлями не будуть працювати на такому контролері. Має в своєму архіві багату кількість функцій для налаштування мережі. Також присутній в нього достатньо простий і зручний графічний інтерфейс, який дає змогу поближче ознайомитись з запущеною мережею, отримати необхідні відомості.

Далі і останнім у черзі йде мережевий елемент – *хост*.

Дуже цікава назва у даного елемента. Адже якщо перевести напряму з англійської, то це буде хтось, хто приймає гостей. У ролі хоста може виступати будь-який пристрій, який впроваджує сервіси типу “клієнт-сервіс” в режимі серверу по будь-яким інтерфейсам. Тобто це може бути будь-який комп’ютер, який відключений до мережі. Наприклад: ПК(персональний комп’ютер), ноутбук, сервер, навіть у ролі хоста може виступати принтер, який під’єднаний до мережі.

Після вищерозглянутих питань видно, що SDN особливо підходить для реалізації ЦОД з потужною функціональністю, яка підтримує централізоване керування мережею, розгалужену переадресацію, розгортання віртуальних машин. Тобто можна сказати, що технологічний тренд в сторону хмарних рішень на базі SDN визначає все майбутнє мереж ЦОД.

2.3. Шляхи побудови кластерної мережі центру обробки даних на базі SDN

Розглянемо які ж популярні методи реалізації традиційних (фізичних) дата центрів на базі технології SDN існують та використовуються в повсякденні:

Серед доступних технологій SDN широке визнання отримала методологія використання SDN-контролера і накладеної мережі для забезпечення належного рівня переадресації та поділу управління. Ця техніка передбачає централізовану доставку політик через SDN-контролер, можливість віртуалізувати мережеві ресурси і гнучко їх планувати. Віртуалізація інфраструктурної мережі ґрунтується на технологіях оверлею, таких як «віртуальна розширена LAN» (VXLAN). Оверлей-протокол звільняє сервіси та ресурси від обмежень, пов'язаних з їх фізичним розташуванням і

дозволяє створити велику логічну мережу рівня 2 для спільного використання ресурсів в центрах обробки даних.

VXLAN розширює L2-мережі за допомогою інкапсуляції MAC-in-UDP і використовує переваги L3-мереж для розгалуженого балансування навантаження, високої надійності, що збагачує можливості L2-мереж. Вузли входу і виходу тунелю VXLAN називаються кінцевими точками віртуального тунелю (VTEP), вони інкапсулюють і декапсулюют пакети VXLAN.

В процесі еволюції SDN, програмно-конфігуровані і традиційні мережі будуть якийсь час співіснувати завдяки різних бізнес-причин. Яким же чином традиційна ЦОД мережа може плавно перетворитися на SDN?

Розглянемо перший варіант. Це побудова нових SDN-точок всередині існуючого ЦОД.

Слід зазначити: в усіх методах реалізації використовуються терміни шлюз, шлюзовий комутатор. Тому дамо визначення терміну шлюз. Шлюз в мережі – апаратний пристрій для сполучення різних мереж, які, в свою чергу, використовуються різні протоколи обробки даних. Такий пристрій конвертує протоколи одного типу мережевого середовища (фізичне, віртуальне) у інший тип мережевого середовища. У всіх нижчезазначених способах, такі шлюзи, шлюзові пристрої стоять на входах/виходах у кожній мережі аби конвертувати протоколи одного середовища в інше і забезпечити зв'язок між різними типами мереж.

В даному випадку реалізації мережа ЦОД може поступово перетворюється в SDN шляхом побудови нових SDN-точок доставки в існуючих ЦОД при наявності наступних умов:

До початку трансформації традиційних ЦОД в ЦОД на базі SDN, замовники хочуть переконатися в життєздатності SDN-ЦОД шляхом побудови нової гнучкої точки доставки на основі SDN та перевірки нових сервісів через цю точку. Сервіси в нових точках доставки не вимагають взаємодії з традиційними точками доставки.

Замовники будують одну або кілька гнучких точок доставки на основі SDN і запускають на них нові сервіси, або мігрують на нові точки доставки якісь вже з існуючих сервісів. Нові точки доставки вимагають забезпечення L3-комунікації з існуючими точками доставки через основні пристрої ЦОД. Між новими і старими точками доставки не потрібна наявність L2-з'єднання.

Візуально макет такого способу реалізації виглядатиме наступним чином:

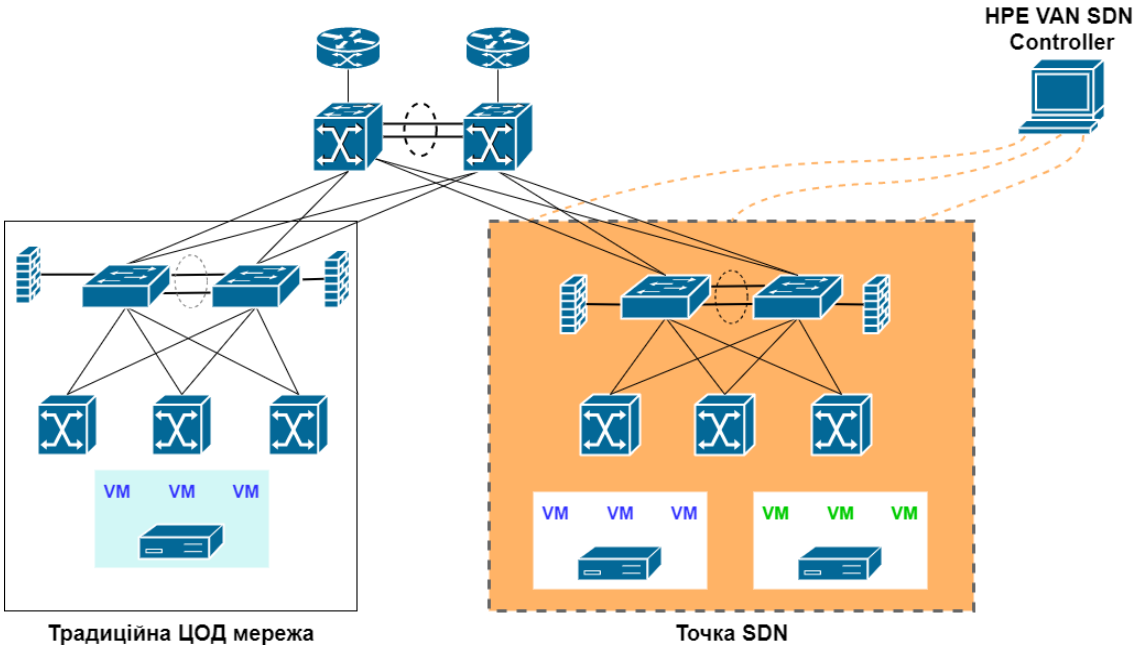


Рисунок 2.9 – Метод побудови ЦОД за допомогою створення SDN-точок всередині існуючої мережі ЦОД

Тепер розглянемо сам процес передачі трафіку з традиційної ЦОД мережі до SDN-точки.

1) Спочатку L3-шлюз на традиційній точці доставки переадресує трафік на основний комутатор на підставі таблиці маршрутизації.

2) Далі основний комутатор пересилає трафік на шлюзовий комутатор на SDN – точці доставки.

3) Після чого шлюзовий комутатор знаходить відповідний маршрут і переадресує трафік на фаєрвол.

4) Фаєрвол в свою чергу знаходить відповідний маршрут і переадресує трафік системі віртуальної маршрутизації і пересилання (VRF), до якої і відноситься хост призначення.

5) Шлюзовий комутатор знаходить таблицю потоків VXLAN для хоста призначення, інкапсулює пакети в формат пакетів VXLAN і потім відправляє їх на VTEP призначення.

6) VTEP (кінцева точка віртуального тунелю) призначення декапсулює пакети VXLAN і переадресовує вихідні пакети на хост призначення на підставі інформації про відповідну MAC-адресу.

Візуально сеанс переадресації виглядатиме, всі шляхи, які вище описані, позначені відповідними позначками:

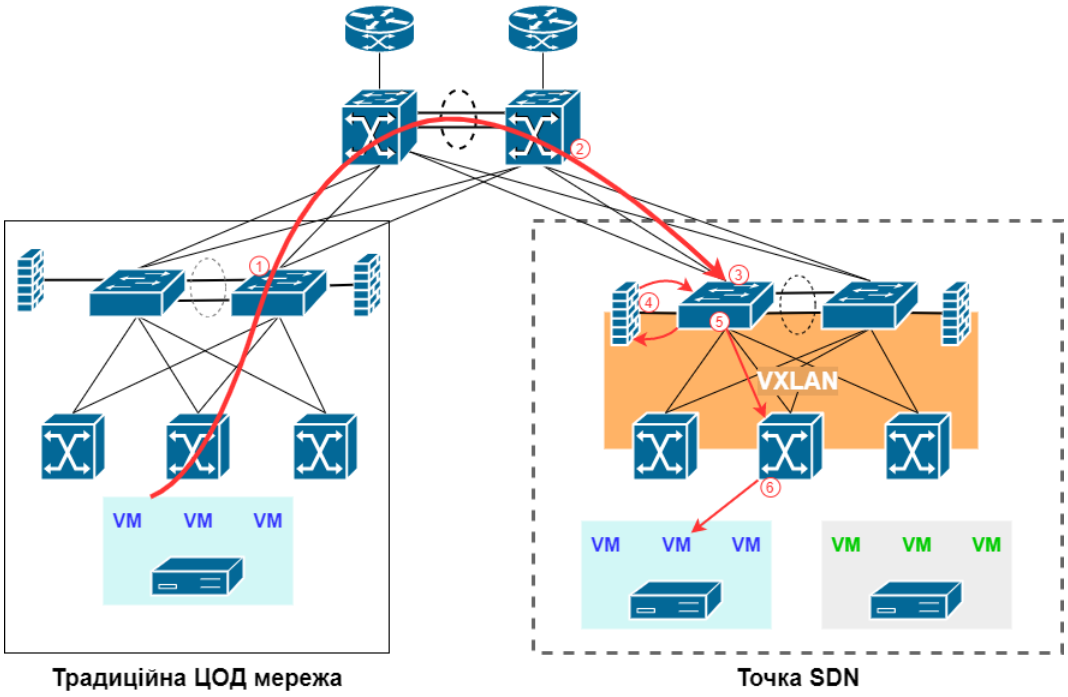


Рисунок 2.10 – Процес переадресації пакетів у першому варіанті реалізації

Під другим варіантом реалізації розуміється розгортання SDN на нових пристроях для проектів розширення традиційних точок доступу.

Цей сценарій найкраще підходить для наступних випадків:

- Замовники хочуть розширити існуючі точки доступу і впровадити в них технологію SDN, не чіпаючи існуючу мережу в точках доступу. Розширення SDN-мережі вимагає тільки L3-з'єднання з існуючою мережею.
- Замовники хочуть мігрувати обчислювальні вузли з існуючої мережі на нові пристрої SDN-шлюзів і реалізувати L2-комунікацію між вихідної мережею і SDN. Прикладом такої ситуації може служити розширення кластера існуючої мережі.

Розглянемо спочатку випадок побудови L3-комунікації. В ньому існуюча традиційна мережа і SDN підключаються до основного вузла ЦОД через агрегуючі комутатори. Під агрегуючими комутаторами розуміють комутатори, які поєднують традиційні комутатори доступу з опорною мережею по волоконно-оптичним лініям зв’язку.

Агрегуючі комутатори двох мереж можуть бути з'єднані безпосередньо для більшої ефективності переадресації трафіку і зниження навантаження на основний вузол.

Візуальна схема побудови такої мережі зображена нижче:

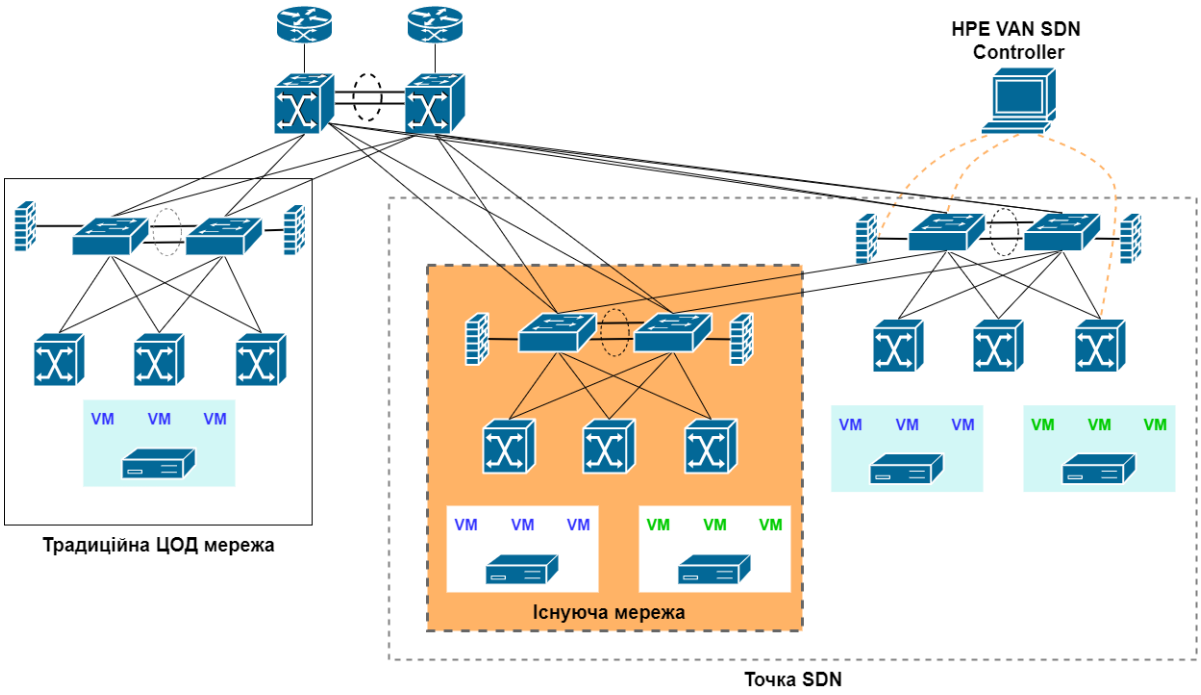


Рисунок 2.11 – Метод побудови ЦОД за допомогою розгортання SDN на нових пристроях

Переадресація виконується в даному варіанті реалізації наступним чином:

1) L3-шлюз традиційної мережі переадресує трафік на шлюзовий комутатор в SDN-мережі на підставі таблиці маршрутизації.

2) Шлюзовий комутатор знаходить відповідний маршрут і переадресує трафік на фаєрвол.

3) Фаєрвол знаходить відповідний маршрут і переадресує трафік на VRF, до якої відноситься хост призначення.

4) Шлюзовий комутатор знаходить таблицю потоків VXLAN для хоста призначення, інкапсулює пакети в формат пакетів VXLAN і потім відправляє їх на VTEP призначення.

5) VTEP призначення декапсулює пакети VXLAN і переадресовує вихідні пакети на хост призначення на підставі інформації про відповідну MAC-адресу.

Візуально процес переадресації набуватиме наступного вигляду:

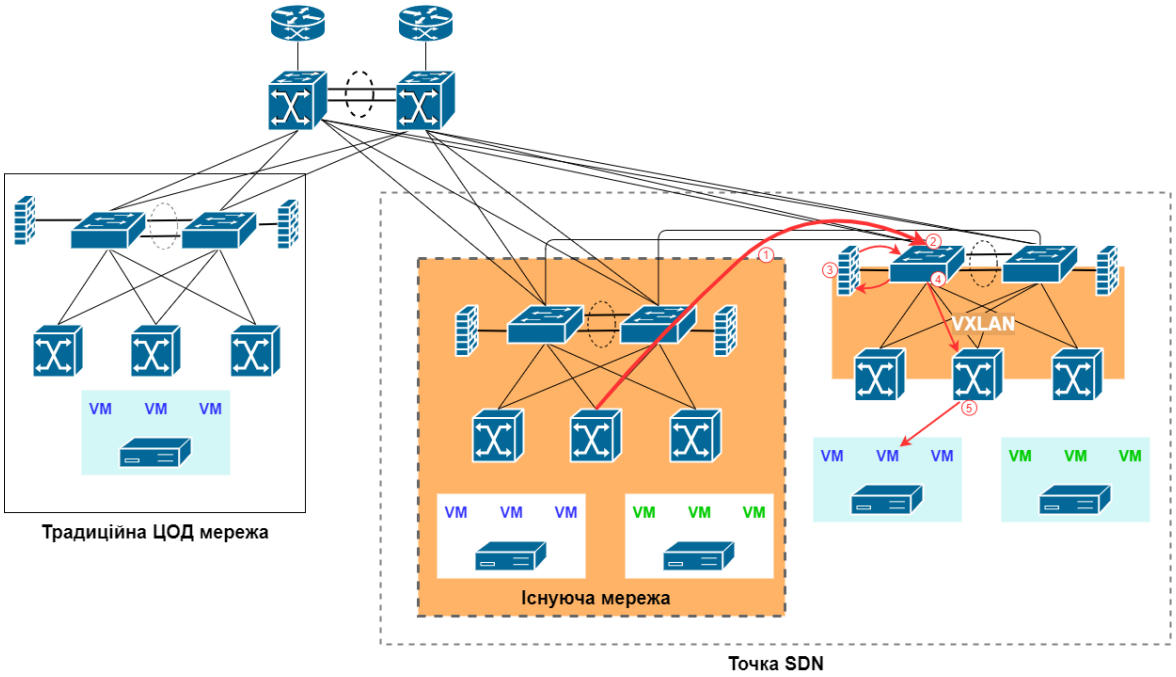


Рисунок 2.12 – Процес переадресації пакетів у другому варіанті реалізації

Також існує варіант даної реалізації але через L2-комунікації.

В даному випадку традиційна мережа підключається до L2-мосту в новій SDN-мережі. L2-мости зіставляють ідентифікатори VLAN, використовувані в традиційній мережі, з VNI, використовуваними в SDN-мережі, встановлюючи між двома мережами L2-з'єднання. L3-шлюз для хостів традиційної мережі може мігрувати на шлюзові комутатори в SDN-мережі. Розширені SDN-точки доставки підключаються до основного вузла комутатора ЦОД через нові шлюзові комутатори.

Візуально схема реалізації побудови мережі виглядатиме:

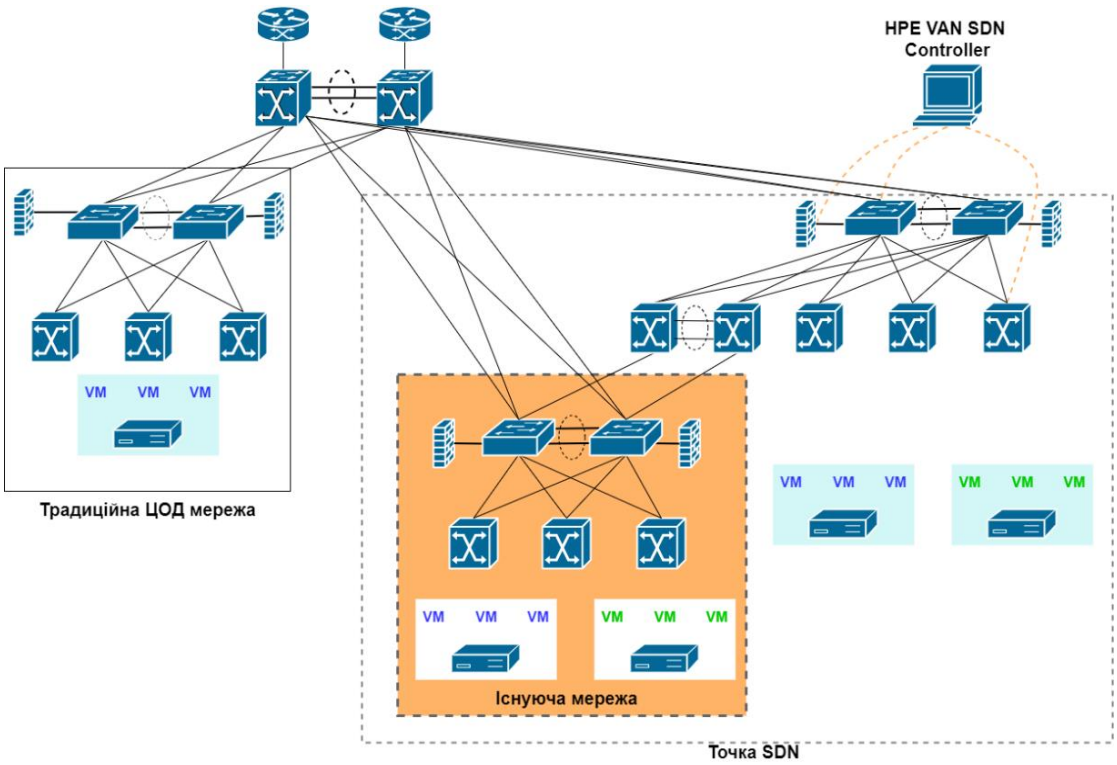


Рисунок 2.13 – Метод побудови ЦОД за допомогою розгортання SDN на нових пристроях через L2-комунікації

Переадресація в такій структурі/мережі виконується наступним чином:

1) Агрегуючий комутатор традиційної мережі переадресовує L2-трафік на L2-міст в SDN-мережі через канал Eth-Trunk.

2) L2-міст інкапсулює формат пакетів VXLAN на основі підготовленого контролером маппінга VXLAN-VNI, знаходить хост призначення в таблиці потоків VXLAN і переадресовує пакети на VTEP призначення.

3) VTEP призначення декапсулює пакети VXLAN і переадресовує вихідні пакети на хост призначення на підставі інформації про відповідний MAC-адресу.

Візуально процес передачі виглядатиме наступним чином:

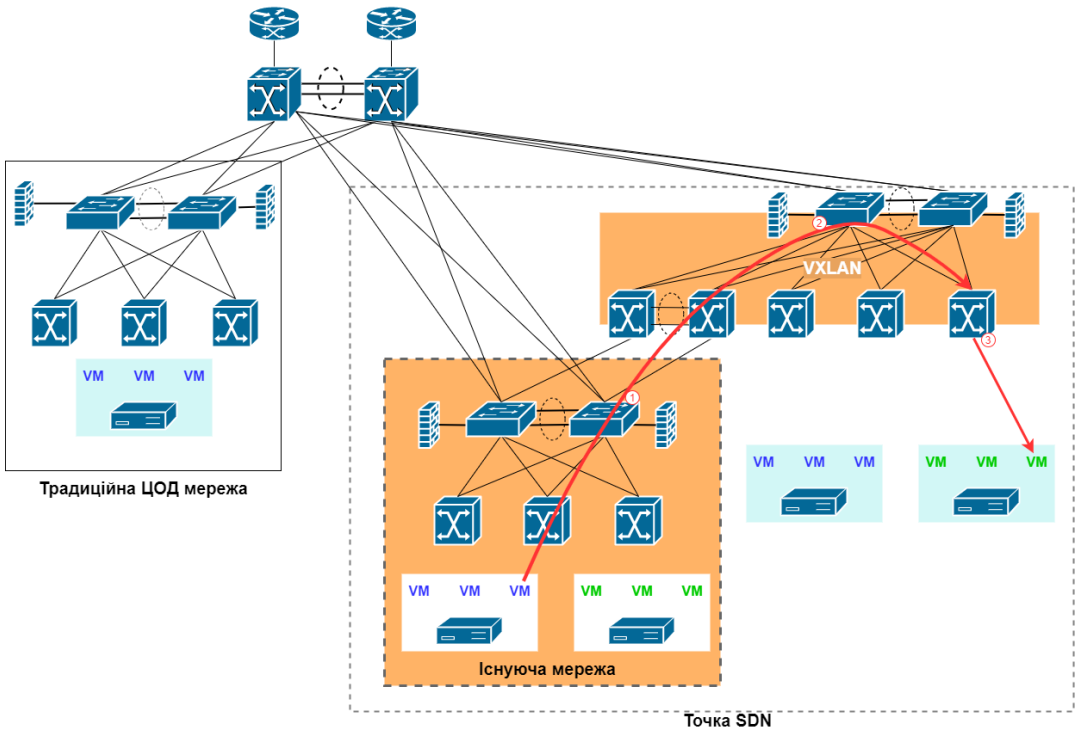


Рисунок 2.14 – Процес переадресації пакетів у третьому варіанті реалізації

Eth-trunk: це агрегація двох мережевих портів. Це еквівалентно протоколу, і основною функцією цього протоколу є збільшення трафіку, наприклад, якщо агрегується два мережевих порти, в результаті виходить вдвічі більше трафіку.

Trunk port - це канал типу «точка-точка» між комутатором і іншими мережевими компонентами.

Інкапсуляція пакетів - це процес передачі даних з найвищого рівня стеку протоколів до найнижчого (фізичного) рівня, щоб бути переданими фізичному середовищі (кручена пара, оптичне волокно, Wi-Fi). На кожному рівні кожен протокол додає до даних, що передаються певну частину власної інформації.

Декапсуляція пакетів це обернена дія до інкапсуляції. Тобто коли інформація проходить з найнижчого рівня стеку протоколів до найвищого і в процесі обробки пакету прибираються певні частини інформації, в залежності від протоколу.

Virtual Extensible LAN (VXLAN) - це технологія для мережевої віртуалізації. Вона, в свою чергу, була створена для вирішення певного ряду проблем, щодо масштабування в великих обчислювальних системах. Дана технологія використовує дуже подібну з VLAN техніку MAC інкапсуляції у UDP пакетах.

Підсумок щодо методів реалізації ЦОД на базі SDN: Мережі ЦОД швидко розширюються, розповсюджуються, щоб забезпечити стрімко зростаючі потреби сервісів та незчисленно великої кількості користувачів. В зв'язку з багатьма прогнозами: до 2020 року в сфері побудови мереж ЦОД в усьому світі очікувалось майже десятикратне зростання.

Так і сталося. І розвиток мереж ЦОД продовжує рости вгору, за допомогою віртуалізації та сучасних рішень SDN. Розвиток ЦОД вимагає також і прогресу в керуванні мережевими ресурсами. Галузеві тренди показують, що технологія SDN перетворюється в основу для сучасних ЦОД мереж. Тому раніше операторів або компаній-замовників турбували питання на кшталт: як можна модифікувати мережу, підключити нові функції, розширити мережу без зайвих інвестицій та проблем. Ці питання допомагає вирішити концепція побудови/переходу мереж на технологію SDN. Оскільки можливо розширити мережу виділивши під це додаткові віртуальні хости, віртуальні ресурси. Нові функції досить швидко стають доступними на програмно-конфігурованих мережах, достатньо їх зібрати у вигляді додатку і підключити до центрального мережевого контролеру. А модифікувати мережу можливо завантаживши необхідні завдання, політики до контролера, який в свою чергу завантажить необхідні політики на необхідні мережеві пристрої.

Тому зараз більшість компаній працюють над вирішенням питання: як забезпечити перехід з традиційних мереж на SDN-рішення, мінімізувавши вплив, зміни існуючих сервісів, функцій і обсяг необхідних вкладень (інвестицій).

2.4. Методи визначення характеристик функціонування кластерної мережі ЦОД

Насправді, це питання можливо вирішити, запустивши на відповідних платформах, сервісах симуляцію власного макету мережі і провести тести продуктивності мережі, аби дізнатися чи буде відповідати віртуальна мережа очікуванням.

Процес симуляції мережі – це процес відтворення роботи мережі, максимально наближеної до реальної, на віртуальному обладнанні (таке обладнання являється реплікою фізичного обладнання і виконує відповідно майже ті самі задачі, що і фізичне обладнання). Але на деяких платформах існують можливості підключити інші сторонні пристрої (віртуальні або навіть реальні).

Варіант симуляції мережі, який був вибраний для розгляду в даній роботі, являється найдоступнішим та найпопулярнішим продуктом в цілях

академічного дослідження. Це побудова мережі на базі безкоштовної платформи з відритим програмним кодом Mininet.

2.5. Висновок до другого розділу

В даному розділі були розкриті питання, щодо кластерних мереж, їх призначення та використання. Також були розглянуті технології віртуалізації та кластеризації, приклади їх використання. Були представлені та пояснені реальні методи побудови ЦОД мереж на базі SDN мереж. Були розкриті питання щодо SDN мереж, їх особливості і переваги, а також були розібрані їх основні мережеві компоненти.

Як висновок, можна сказати, що кластерні мережі реалізуються об’єднанням мережевих елементів по групам (кластерам), що надає більш високу надійність мережі та доступність послуг, що надаються. Адже задачі між елементами у кластері розподілені так, аби при відмові певного вузла, інші два вузла могли б розподілити між собою його задачі і виконувати, допоки відмовлений вузол знову не почне працювати. Саме технологія кластеризації і забезпечує функції розподілу задач у мережі за певним алгоритмом.

Технологія віртуалізації дозволяє відтворити фізичний пристрій (та його роботу) чи процес у віртуальному (цифровому) вигляді, використовуючи обчислювальні ресурси комп’ютера для такої симуляції.

Мета мереж SDN полягає у тому, аби перенести керування мережею на певний фізичний, або віртуальний пристрій (частіше за все це контролер). Мережеві комутуючі пристрої керуються контролером. Такий контролер за допомогою встановлених додатків має змогу обчислювати для кожного комутуючого пристрою найвигідніший шлях переадресації трафіку при тій чи іншій умові. Тобто за допомогою SDN мереж та прописаної логіки на контролері можливо автоматизувати конфігурацію елементів мережі та роботи мережі в цілому. А також SDN мережі за підтримки технології віртуалізації дають змогу зекономити на придбанні та налаштуванні обладнання.

Часто мережі SDN можуть складатися частково з фізичного обладнання, частково з віртуального. За допомогою належного налаштування мережевих мостів можна забезпечити належне з’єднанні між фізичною та віртуальною частинами мережі.

Аби не витратити зайвих коштів при переході від традиційної L2-мережі до SDN мереж, можливо використати сервіси для створення імітаційної моделі власної мереж для відтворення, запуску та тестування мережі віртуально для знаходження її недоліків та їх усунення заздалегідь, без попередньої побудови мережі. Також на імітаційній моделі можливо дізнатися технічні характеристики та можливості мережі та її компонентів.

РОЗДІЛ №3. Побудова моделі кластерної SDN мережі ЦОД на базі платформи Mininet

3.1. Опис платформи Mininet. Її основні характеристики та призначення мережевих елементів. Способи її встановлення

Програмний продукт Mininet дає змогу користувачеві створити віртуальну програмно-конфігуровану мережу з віртуальними мережевими елементами, які наближені до реальних, та запустити, протестувати, дослідити мережу на власному комп’ютері.

Mininet будує прототипи віртуальних програмно-конфігурованих мереж (SDN). Тому мережа складається з тих же елементів, що і реальна SDN мережа: віртуальних хостів, віртуальних комутаторів та віртуального контролеру.

Даний програмний продукт активно розвивається та використовується в найкращих вузах світу для практики в галузі проектування та побудови програмно-конфігурованих мереж.

Запущена мережа працює, використовуючи апаратні ресурси хоста, а саме:

- необхідну для роботу мережі кількість ресурсів, яка ніяким чином не обмежена. Якщо віртуальна мережа працює на основному хості (ваш персональний комп’ютер);
- необхідну для роботу мережі кількість ресурсів, яка обмежена ресурсами віртуальної машини, на яку було виділено окрему кількість апаратних ресурсів з основного хоста.

Платформа Mininet може бути завантажена та встановлена наступними шляхами:

Встановлена на ОС Linux. Для цього слід завантажити необхідні для роботи пакети необхідними командами та з них вже встановити Mininet

командами. Отже при завантаженні платформи Mininet ми отримуємо наступні папки: mininet, openflow, pox. Судячи з назви можна зрозуміти, що папка mininet складається з інструментів для роботи усієї платформи в цілому, також в даній папці є утиліта miniedit та багато прикладів для зразку побудови різних мережа та застосування різних функцій. Папка openflow використовується для коректної роботи протоколу openflow на даній операційній системі. Папка pox – це завантажений образ контролера, який у Mininet використовується по-замовчуванню, тут також знаходяться різні інструменті для можливості роботи і запуску віртуальних мереж, з використанням даного контролеру. Ресурсів з поясненнями встановлення даної платформи безліч, а відео напевне ще більше, але найголовнішим ресурсом являється сайт самого Mininet: http://mininet.org/download/#option-2-native-installation-from-source;

Встановлена в якості віртуальної машини на гіпервізорі. Для цього варіанту слід завантажити образ віртуальної машини, знову ж з офіційного сайту Mininet. Та потім імпортувати його до будь-якої програми-гіпервізору (VirtualBox, VMware) з необхідними налаштуваннями. Після чого буде встановлена віртуальна операційна система виду Linux Ubuntu з вбудованими в неї програмними пакетами Mininet. Ресурсом пояснення виступає знову ж таки сайт Mininet, на якому одразу можна завантажити необхідний образ віртуальної машини: http://mininet.org/download/#option-1-mininet-vm-installation-easy-recommended.

Для створення мережі можна використовувати або заготовлені типи топологій мережі (templates), або ж створений файл топології на мові програмування Python. Був доданий варіант створення користувацької мережі, оскільки заготовлені типи (вбудовані в бібліотеки Mininet) мають обмежені можливості і не завжди можуть розкрити весь потенціал ідеї мережі користувача платформи.

При створенні користувацької мережі з’являються можливості залучення різноманітних функцій, шляхом додання до файлу різноманітних бібліотек, які існують на Python.

Оскільки створені на платформі Mininet мережі являються програмно-конфігурованими, тобто SDN, вони підтримують віддалене керування (за допомогою протоколу OpenFlow), яке розташоване в центральному віртуальному контролері. Тому при створенні користувацької топології є можливість під’єднати віддалений мережевий контролер, ввівши відповідну команду та його логічну адресу та порт підключення. Приклад підключення виглядає наступним чином:

```
c0 = net.addController(name='c0',
                       controller=RemoteController,
                       ip='0.0.0.0',
                       protocol='tcp',
                       port=6633)
```

Лістинг коду № 3.1 – Підключення віддаленого контролеру до віртуальної мережі SDN на базі Mininet у файлі топології мережі

Тобто до файлу користувацької топології заноситься змінна контролеру “c0”. Яка далі описується що це не локальний (net) елемент, вказується що це віддалений контролер (вказує платформі, щоб не використовувала контролер по-замовчуванню), його логічна адреса та порт підключення. Але для цього слід не забути підключити функцію мережевого контролера, для цього на початку файлу слід підключити бібліотеку з цією функцією відповідною командою.

Після даних маніпуляцій слід ввімкнути контролер, де би він не розташовувався, та почекати коли він буде в статусі очікування підключення. Після чого слід запустити мережу і Mininet виконає підключення мережі до даного контролера.

Також існує інший спосіб підключення мережі до віддаленого контролера. Для цього слід під час запуску мережі, до команди запуску, слід приписати наступні параметри:

```
sudo mn --controller=remote,ip=192.168.0.103,port=6653
```

Лістинг коду № 3.2 – Запуск найпростішої мережі SDN з підключенням віддаленого мережевого контролеру

В іншому випадку, якщо не вказувати ніяких відомостей про контролер, Mininet використає вбудований (завантажений разом з файлами Mininet) POX контролер.

3.2. Принципи створення імітаційної моделі мережі на базі Mininet. Переваги та недоліки платформи

Які ж існують основні способи побудови мережі на базі Mininet? В першу чергу, слід визначитись, якого типу буде мережа: без віддаленого контролеру, або ж навпаки з ним.

Переглянемо основні існуючі шляхи побудови мереж на Mininet:

Побудова мережі вбудованого (передвстановленого) типу. Для такого запуску мережі слід у терміналі операційної системи прописати: *sudo mn <параметри>*. Параметрами виступає тип топології мережі та різноманітні параметри запуску мережі (обмеження пропускної здатності у мережі, збільшення затримки у мережі).

Вбудовані топології в мережі представляють собою структуру мережі таким чином, аби користувач ввівши певні цифри, отримав готову топологію з певною кількістю мережевих хостів (цифра, яку користувач ввів). Розглянемо які присутні типи вбудованих топологій.

- *mininal*. Використовується за замовчуванням при запуску mn без параметрів. У цьому випадку створюються два хоста, підключених до одного комутатора, який, в свою чергу керується OpenFlow-контролером (про цей пристрій теж поговоримо пізніше). У даній топології не можна задати довільне число хостів або комутаторів. Команда для створення та запуску: *sudo mn*
- *single*. Як і у випадку з *minimal*, всі хости підключаються до одного комутатора. Єдина відмінність - це можливість вказати їх (хостів) кількість.
  Команда для створення та запуску такої мережі виглядає наступним чином: *sudo mn --topo=single,4*
  де 4 – кількість хостів в мережі (див. рис. 3.1)

```
mininet@mininet-vm:~$ sudo mn --topo=single,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Рисунок 3.1 – Приклад запуску топології типу *single* з 4 хостами

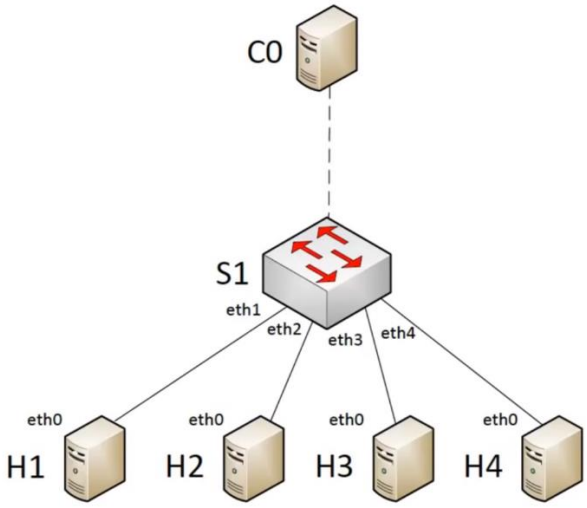


Рисунок 3.2 – Візуальне зображення топології типу *single* з 4 хостами

- *linear*. Топологія описує мережу в якій всі хости підключені до власних комутаторів, які в свою чергу з'єднані між собою.
  Команда для створення та запуску: *sudo mn --topo=linear,4*
  де 4 – кількість хостів, комутаторів в мережі (див. рис. 3.3)

```
mininet@mininet-vm:~$ sudo mn --topo=linear,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s2, s1) (s3, s2) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
```

Рисунок 3.3 – Приклад запуску топології типу *linear* з 4 хостами

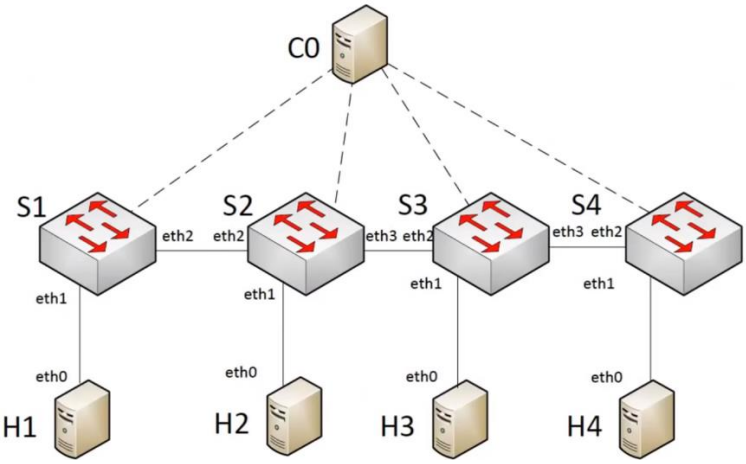


Рисунок 3.4 – Візуальне зображення топології виду *linear* з 4 хостами

- *tree*. Деревоподібна топологія, найбільш складна з перерахованих. Тут в якості параметрів можна вказати глибину ієрархії комутаторів (*depth*) і так само число підключених хостів (*fanout*) до кінцевих (найнижчі в архітектурі) комутаторів.
  Команда для створення та запуску мережі такого типу виглядатиме: *sudo mn --topo=tree, depth=2,fanout=2* (див. рис. 3.5)
  - де *depth=2* – глибина ієрархії комутаторів, в нашому випадку на 2 рівні опуститься ієрархія. Буде 3 комутатори, на першому рівні один комутатор, на другому від роздвоюється і виходить загалом 3.
  - *fanout=2* – кількість хостів, які приходяться на 1 комутатор. Хости додаються лише до комутаторів на найнижчому рівні. В нашому випадку на 2 рівні глибини. Оскільки по 2 хоста на комутатор, то у нас виходить на найнижчому рівні виходить 4 хости на 2 комутатори.

```
mininet@mininet-vm:~$ sudo mn --topo=tree,depth=2,fanout=2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
```
Рисунок 3.5 – Приклад запуску топології типу *tree* з глибиною ієрархії – 2, та кількістю підключених хостів на кожному комутаторі – 2

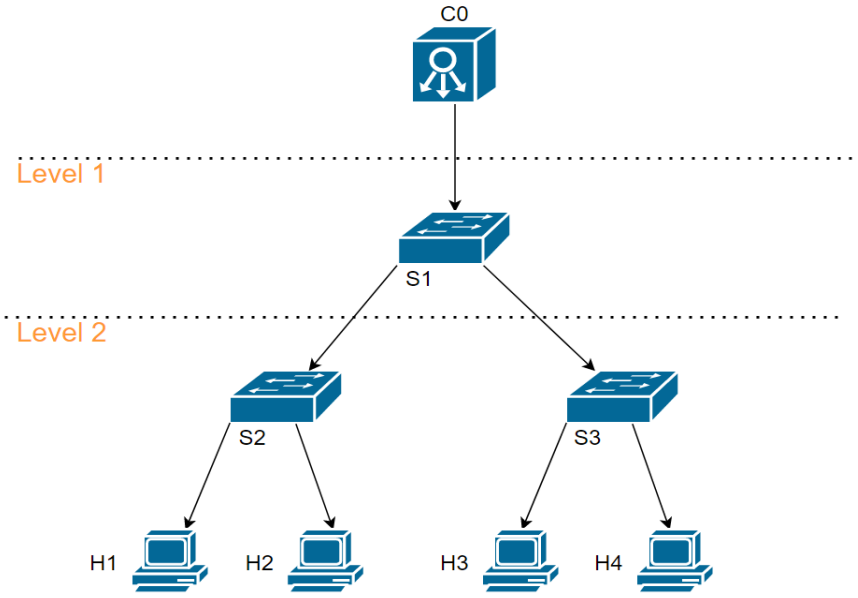


Рисунок 3.6 – Візуальний приклад описаної вище топології

Розглянемо більш складну деревоподібну топологію на основі вище наведеної (*наприклад*):

Якби вказали *глибину(depth)* 3 – тоді б ще на один рівень опустились і в результаті отримали б: на 1 рівні – 1 комутатор, на 2 рівні – 2 комутатори, на 3 рівні – 4 комутатори. Загалом виходить 7 комутаторів.

Кількість хостів на комутатор *(fanout)* не будемо збільшувати, аби не ускладнювати і не заплутувати приклад.

Команда для запуску мережі такої топології виглядала б: *sudo mn --topo=tree, depth=3,fanout=2* (див. рис. 3.7)

```
mininet@mininet-vm:~$ sudo mn --topo=tree,depth=3,fanout=2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6)
 (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
```

Рисунок 3.7 – Приклад запуску топології виду *tree* з глибиною ієрархії - 3 та кількістю хостів на кожному комутаторі -2

Побудова мережі з власною топологією. Для побудови мережі такого типу слід у терміналі операційної системи прописати: *sudo mn --custom= назва_файлу.py <параметри>*. В даному випадку в нас додається запис про те, що використовується користувацька (*custom*) топологія. Та приписується

файл з топологією, яку запускати. В даному файлі прописується створення усіх мережевих елементів, мережевих з'єднань. Параметрами в даній команді можуть виступати задання класу мережі, яка запускається, або ж параметри запуску мережі (обмеження пропускної здатності у мережі, збільшення затримки у мережі).

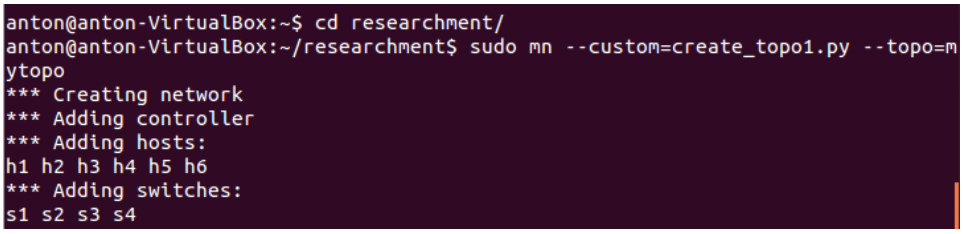


Рисунок 3.8 – Приклад запуску користувацької топології мережі у Mininet

Побудова мережі шляхом запуску файлу-скрипта. Для даного запуску мережі слід написати подібний до вищезазначеного файл з топологією мережі, але слід ще дописати команди, які будуть виконуватись при запуску даного файлу. Запустити файл і мережу відповідно можна командою: *python назва_файлу.py*. Після запуску файлу відбудеться створення та запуск мережі, а також виконання команд, що прописані у файлі.

Усі вищезазначені опції створення мережі були розглянуті без підключенням віддаленого контролера. В такому випадку використовувався класичний завантажений разом з файлами Mininet віртуальний POX контролер.

Якщо розглядати вищезазначені варіанти побудови мереж, але вже з використанням віддаленого контролеру, то для цього достатньо буде приписати до команди запуску мережі наступні параметри: *--controller=remote,ip=192.168.0.103,port=6653*, де вказується, щоб платформа підключила віддалений контролер, який працює за певною адресою та приймає підключення мережевих елементів на певному порті.

Наприклад: аби запустити деревоподібну топологію мережі з підключенням віддаленого контролеру, слід виконати наступну команду: *sudo mn --topo=tree, depth=3,fanout=2 --controller=remote, ip=192.168.0.103, port=6653*.

Під час проведених досліджень та спостережень, були сформовані 3 основні типи побудови мереж з контролером. А саме:

- З вбудованим (по-замовчуванню) контролером. В даному типі віртуальна мережа Mininet під'єднується до локального (вбудованого в Mininet) контролеру POX. Цей спосіб непоганий коли слід швидко запустити мережу і не має вимог щодо управління мережею.
- З віддаленим (зовнішнім) контролером. В даному типі віртуальна мережа Mininet під'єднується до зовнішнього контролеру, використовуючи логічну адресу контролера, який знаходиться поза межами операційної системи з Mininet. Такий спосіб підходить коли варто запустити мережу з використанням власного, досліджуваного контролера. Даний варіант більш професійний, оскільки є можливість конфігурація мережі напряму через контролер.
- З віддаленим-локальним контролером. Цікава назва, але так і є. В даному типі віртуальна мережа Mininet підключена до віртуального контролеру, який розгорнутий на цій же операційній системі, що й Mininet. Цей тип побудови відрізняється від попереднього тим, що мережа під'єднується до контролера, використовуючи локальну адресу: 10.0.2.15 і порт підключення, в залежності від контролеру. Ця адреса резервується платформою Mininet для підключення контролеру та запуску мережі.

У віртуальних контролерів присутня багата кількість портів. Головні та найвідоміші з яких: порт підключення мережевих пристроїв (комутаторів), порт підтримки контролера та порт графічного web-інтерфейсу.

Тепер розглянемо основні переваги та недоліки програмного продукт Mininet. Перевагами тут виступають:

- швидкий запуск найпростішої мережі (командою *sudo mn*);
- підтримка різноманітних функцій у мережі, що зберігаються і активують підключенням бібліотек Python у конфігураційному файлі мережі (файл топології мережі);
- підтримка підключення сторонніх контролерів, комутаторів, маршрутизаторів;
- присутня гнучкість при побудові топології мережі;
- можливість тестування та моніторингу мережевих параметрів;
- присутність графічного інтерфейсу побудови мережі, який під час складання графічного макету мережі перероблює дану мережу у код.

В якості графічного інтерфейсу побудови мережі виступає утиліта Miniedit, яка завантажується разом з файлами Mininet. Запускається така утиліта за допомогою інструментів Python через термінал. Для цього слід у терміналі перейти до папки, де знаходиться Miniedit та прописати: *sudo python miniedit.py*. Після чого відкриється наступне вікно:

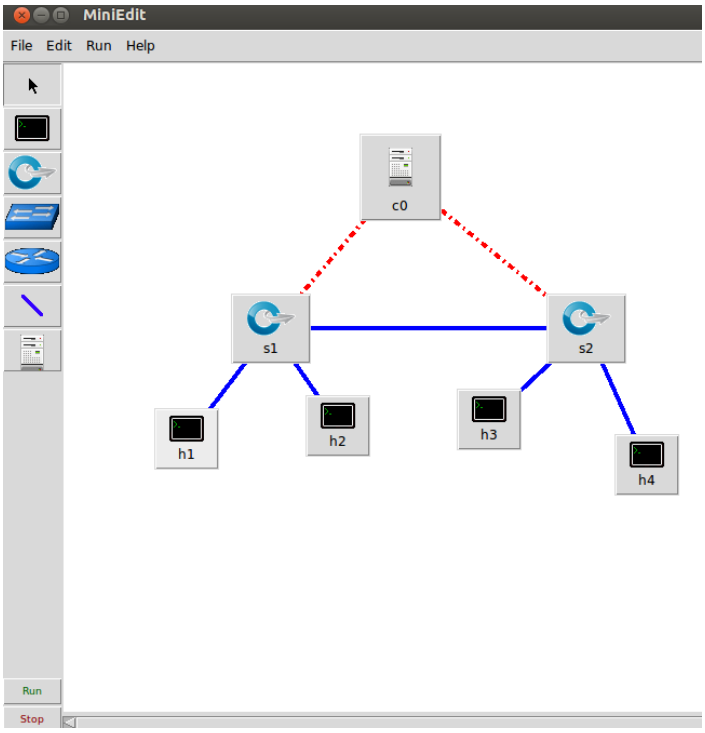


Рисунок 3.9 – Графічний інтерфейс програми Miniedit

Бачимо, що це так званий “конструктор”, де можна візуально побудувати мережу з доступних елементів: хостів, OVS (комутатори з підтримкою OpenFlow), віртуальних звичайних комутаторів (комутатори 2-го рівня), маршрутизаторів, контролерів та все це поєднати віртуальними лініями зв’язку. Коли елементи розташовані на робочому полотні – можна їх налаштувати, для цього натискаємо на кожен елемента та вибираємо: “Properties”. Наприклад, в налаштуваннях віртуального вузла (хоста) можливо: налаштувати логічну адресу, перейменувати пристрій, додати зовнішні інтерфейси, додати VLAN інтерфейси. Важливим моментом являється налаштування контролеру, для цього також переходимо до налаштувань і вносимо наступні відомості: робочий порт комутатора, тип контролеру (рекомендовано вибирати зовнішній контролер – Remote Controller), та унікальна логічна адреса контролеру.

Після побудови та налаштування мережі можливо мережу одразу запустити, натиснувши на кнопку “Run”. Також можливо експортувати дану топологію Mininet мережі до Python файлу. Спосіб з експортуванням коду являється рекомендованим, оскільки в разі проблем можливо відредагувати код у файлі і запустити ще раз, у способі з запуском напряму з Miniedit виникають труднощі і доводиться перезапускати утиліту.

Для експорту слід на панелі задачі вибрати *File – Export Level 2 Script* та вибрати місце збереження файлу. Можна переглянути код, який згенерувався для нашої топології, для цього відкриємо файл з топологією мережі у будь-якому текстовому редакторі і побачимо наступну картину:

*example_miniedit1.py ×

```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

def myNetwork():

    net = Mininet( topo=None,
                   build=False,
                   ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                         controller=RemoteController,
                         ip='127.0.0.1',
                         protocol='tcp',
                         port=6633)

    info( '*** Add switches\n')
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)


    info( '*** Add hosts\n')
    h3 = net.addHost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
    h2 = net.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
    h4 = net.addHost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)
    h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)

    info( '*** Add links\n')
    net.addLink(h1, s1)
    net.addLink(s1, h2)
    net.addLink(h3, s2)
    net.addLink(h4, s2)
    net.addLink(s2, s1)

    info( '*** Starting network\n')
    net.build()
    info( '*** Starting controllers\n')
    for controller in net.controllers:
        controller.start()

    info( '*** Starting switches\n')
    net.get('s1').start([c0])
    net.get('s2').start([c0])

    info( '*** Post configure switches and hosts\n')

    CLI(net)
    net.stop()
```

Рисунки 3.10.1 – 3.10.2 – Лістинг коду користувацької мережевої топології

Бачимо структуру, щодо побудови мережі, яку можна розділити на частини: прив’язка необхідних для роботи python бібліотек, задання класу мережі (створення мережі), додання до мережі послідовно мережеві елементи та віртуальні зв’язки між елементами, створення алгоритму виконання скрипту (створити мережу з елементами, ввімкнути контролер, ввімкнути комутатори).

Після чого можливо запустити користувацьку мережу, використовуючи інструменти Python, а саме, прописавши: *sudo python example_miniedit1.py*. Після чого почнеться створення мережі, якщо все прописано правильно:

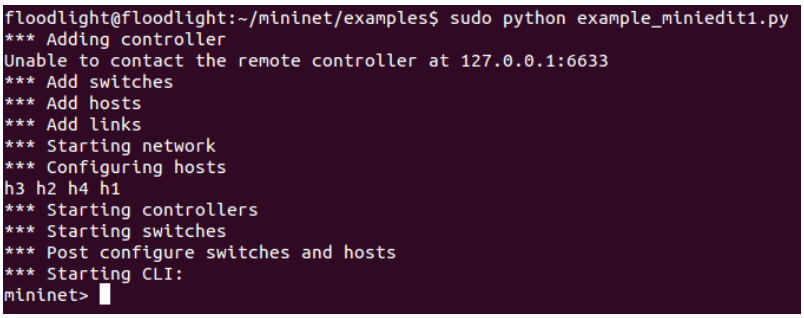


Рисунок 3.11 – Приклад запуску мережі, створеної на Miniedit

Недоліками в платформі Mininet виступають випадки:

- якщо віртуальна мережа на Mininet підключається до контролеру – ніяка більше мережа не зможе використати його для підключення, допоки підключена мережа не припинить з ним роботу;
- коли запускається віртуальна мережа на Mininet, то усі адаптери на операційній системі, де працює мережа, вимикаються і працює лише один адаптер з адресою: 10.0.2.15. І можуть виконуватись операції передачі пакетів лише усередині мережі Mininet;
- ресурси, що передаються усередині мережі Mininet (трафік) увесь віртуальний. І коли були спроби перевантажити мережу (перевищити допустиму пропускну здібність) платформа не давала змогу це зробити і кожен раз підганяла відісланий трафік під пропускну здібність.
- як вже раніше згадувалось, що в мереж ЦОД можливе підключення одного мережевого вузла (серверу) до декількох комутаторів одночасно. В
- коли віртуальна мережа, що створена та запущена на Mininet, не має доступу до елементів, що знаходяться не в межах мережі. Тобто віртуальний вузол мережі Mininet на одній віртуальній машині, не зможе доставити пакети на інший віртуальний вузол мережі Mininet на іншій віртуальній машині, навіть якщо змінити логічні адреси вузлам.

Були спроби вирішити останню проблему. Для цього була наступна задумка: з віртуального вузла мережі Mininet підключитися до FTP клієнту, який налаштований на головному хосту, та завантажити якийсь файл. Для підготовки, аби віртуальний вузол мав хоч якийсь доступ поза мережею, було створено віртуальне підключення, а саме: адаптер 127.0.0.1 (підключення, яке пов’язує головний хост і віртуальну машину) був підключений напряму до адаптеру 10.0.2.15 (адаптер Mininet мережі, який вмикається при запущеній мережі). Тобто дане підключення активувалось коли мережа

Mininet була запущена. Після чого, за допомогою команди на віртуальному хості було виконано підключення до FTP клієнта, у якого є унікальна логічна адреса. Статус підключення був позитивний. Далі, на тому ж хості генерувався запит на завантаження файлу і відправлявся на FTP-клієнта.

Після чого були наступні статуси: “Запит надіслано”, “Отримання файлу”, “Помилка в отриманні файлу”. І так кожний раз.

Тому було зроблено висновок, що ми хоч і посторонім шляхом відкрили доступ з мережі, але до мережі все одно доступу немає.

3.3. Способи дослідження мережевих параметрів на платформі Mininet

Коли ми вже знаємо способи створення мережі на Mininet, слід визначити, які ж параметри можливо досліджувати на платформі Mininet. Усі параметри досліджувались на найпростішій мережі Mininet. Оскільки було проведено багато дослідів над різними мережами, тому тут зібрані найактуальніші та найкорисніші дослідження мережевих параметрів, а саме:

- *Перевірка досягаємості* (доступності) мережевих вузлів, за допомогою інструментів ping, pingall.
  За допомогою системного інструменту ping ми можемо відправити перевірочний пакет з одного вузла на інший, для визначення доступності вузлів відносно один одного. По-замовчуванню відправляється одразу 3 пакети, але дану команду можна модифікувати, додавши різні параметри, на кшталт: *ping -c 20 -s 500 -i 5 ip_отримувача* – передача 20 пакетів розміром 500 байт з інтервалом 5 секунд один від одного на адресу отримувача.
  За допомогою інструменту pingall, кожний вузол відправляє перевірочний пакет на усі інші вузли, а ті, в свою чергу, відправляють на нього перевірочний пакет, після даної перевірки кожен вузол звітує результат. Якщо пакет дійшов до отримувача, то буде відображатись наступна ситуація: h1 -> h2, h2 -> h1, якщо не дійшов: h1 -> x, h2 ->x. Тобто, якщо пакет дійшов до отримувача і навпаки, це означає, що даний відрізок зв’язку готовий до роботи, вузли “бачать” один одного і можуть передавати трафік.
  За допомогою команди *pingallfull* можливо дізнатися більш розширену статистику по кожному відрізку зв’язку. А саме: параметр RTT, мінімальний, середній та максимальний час передачі пакетів на відрізку.

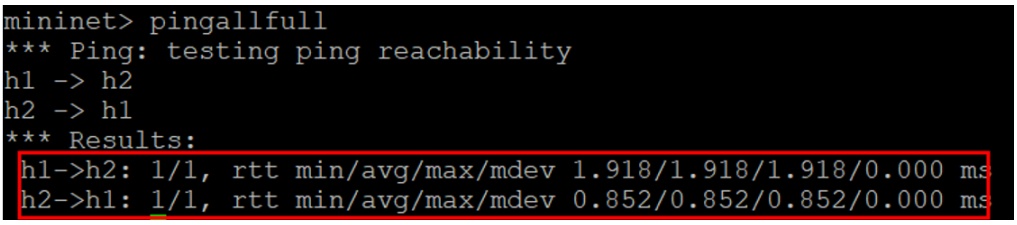


Рисунок 3.12 – Приклад перегляду доступності мережевих елементів за допомогою інструменту pingallfull

- *Перевірка пропускної здібності* між усіма мережевими вузлами, за допомогою інструменту iperf. При запуску команди iperf обчислюється значення найменшої та найбільшої пропускної здібності по усій мережі по протоколу TCP, за час даної сесії роботи мережі.

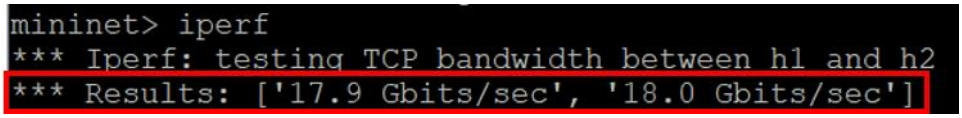


Рисунок 3.13 – Приклад перевірки пропускної здібності у мережі за допомогою інструменту iperf

- Також існує можливість запустити за допомогою інструменту iperf на мережевих вузлах дослідження емуляції *передачі TCP* або *UDP трафіку*, а також його відстеження . Для цього слід відкрити термінали мережевих вузлів, прописавши: *xterm h1 h2* – відкриє термінали першого та другого хостів. Та далі на одному хості прописати що це сервер, а на іншому, що це клієнт і вказати логічну адресу серверу, до якого він буде надсилати запити.

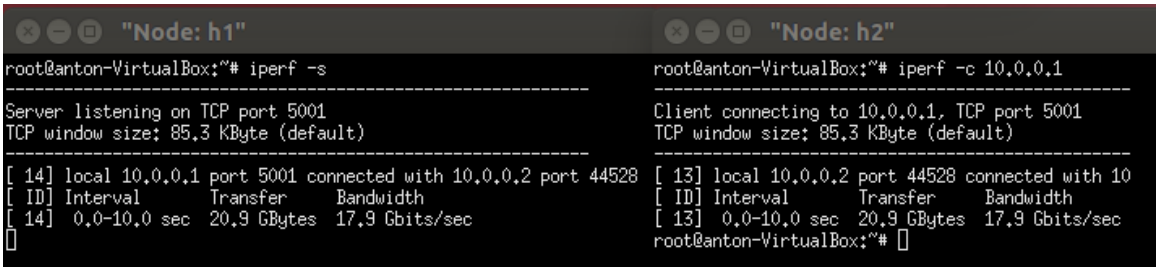


Рисунок 3.14 – Приклад відстеження TCP трафіку
за допомогою інструменту iperf

Аби дослідити передачу та відстеження за UDP протоколом слід також відкрити термінали на 2 вузлах і прописати на одному вузлі що він udp-сервер, а на іншому що він UDP-клієнт і вказати логічну адресу udp-серверу, до якого будуть надсилатись запити.

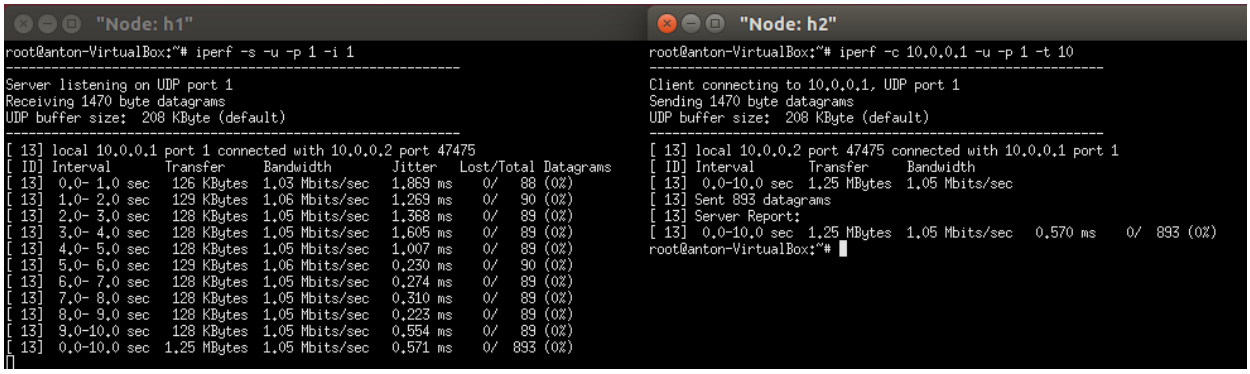


Рисунок 3.15 – Приклад відстеження UDP трафіку
за допомогою інструменту *iperf*

- Відстеження трафіку на інтерфейсі, за допомогою інструменту *tcpdump*. Даний інструмент також краще запускати на мережевих вузлах, через термінал. Для цього слід відкрити термінали мережевих вузлів, прописавши: *xterm h1 h2*. Та прописати на 1-му вузлі: *tcpdump h1-eth- -nn* – відстеження трафіку на інтерфейсі eth0 на 1-му вузлі.

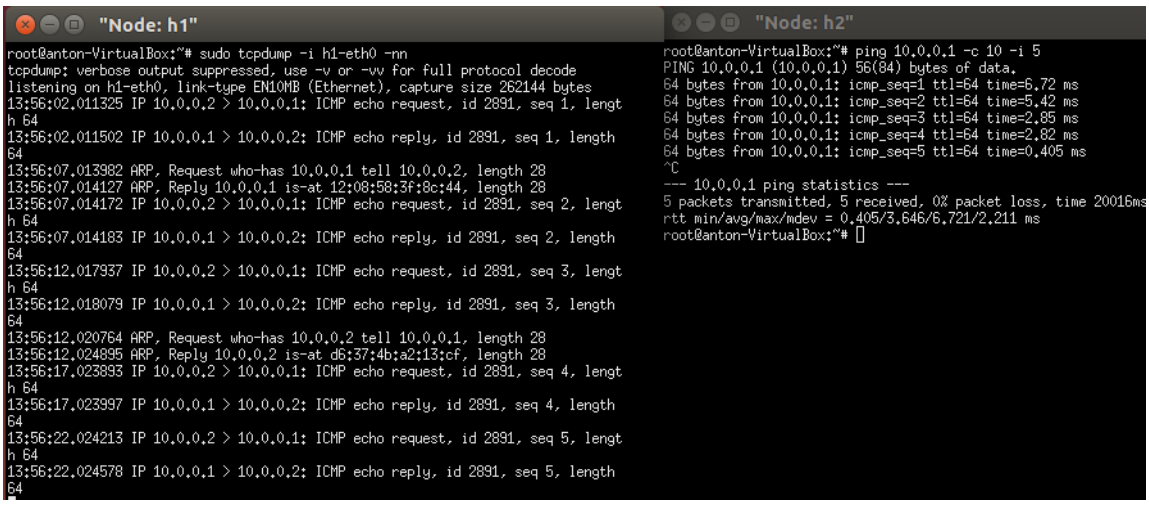


Рисунок 3.16 – Приклад відстеження трафіку через інструмент tcpdump

Тепер розглянемо ще невелику особливість віртуальної платформи Mininet, перед тим, як перейти до практичної частини роботи.

Досліджувана платформа підходить для вивчення та дослідження сніферів. Сніферами являються програми, які перехоплюють (в гарному сенсі) та аналізують мережевих трафік на всіх активних мережевих інтерфейсах на операційній системі. Наприклад: в мережі можна ввімкнути сервер (підняти веб-сервер на мережевому вузлі) та згенерувати на нього трафік (емуляція заявок в реальній мережі). Після чого можна використати інструмент "*tcpdump*" чи утиліту Wireshark для зняття параметрів трафіку та детального аналізу трафіку. Такий приклад роботи сніферів майже нічим не буде відрізнятися від роботи сніферів в реальній фізичній/віртуальній мережі.

3.4. Висновки до третього розділу

Mininet являється безкоштовною платформою для імітування роботи віртуальних мереж з підтримкою технології SDN. Можна Mininet завантажити у вигляді окремої віртуальної машини, або ж завантажити пакети для роботи на комп'ютер с ОС Linux. У Mininet вбудовано декілька типів різноманітних топологій. На даній платформі є можливість створювати власні мережі будь-яких топологій і підключати до мережі пакети Python з різноманітними функціями. Також є можливість до таких мереж підключити віддалений контролер, та перенести задачу керування мережею на нього. Також є можливість досліджувати мережеві параметри мережі на Mininet, використовуючи вбудовані в ОС Linux інструменти.

В даній платформі присутній вбудований конструктор мереж, який можна запустити, використовуючи інструменти Python, і візуально зібрати власну мережу, яка в подальшому конвертується в скрипт на мові Python і може бути запущена безпосередньо за допомогою інструментів мови Python.

РОЗДІЛ №4. Дослідження та побудова мережевого макету кластерної SDN мережі ЦОД на базі платформи Mininet

4.1. Постановка задачі для побудови кластерної ЦОД мережі SDN на платформі Mininet. Встановлення необхідних компонентів. Побудова підготовчих макетів

Як вже на початку говорилося, що задачею стоїть перебудувати кластерну мережу ЦОД під віртуальну SDN мережу, використовуючи платформу Mininet для емуляції макету мережі.

Для реалізації такого макету мережі на базі платформі Mininet було прийнято рішення розділити кластерну мережу ЦОД на дві віртуальні частини (мережі), а саме: користувацьку (для замовників, клієнтів та адміністраторів ЦОД), та мережу дата-центру (сервери), яка і обчислює, обслуговує вхідні запити, запити клієнтів.

Візуально віртуальну мережу, яку слід відобразити на платформі Mininet можна спостерігати нижче:

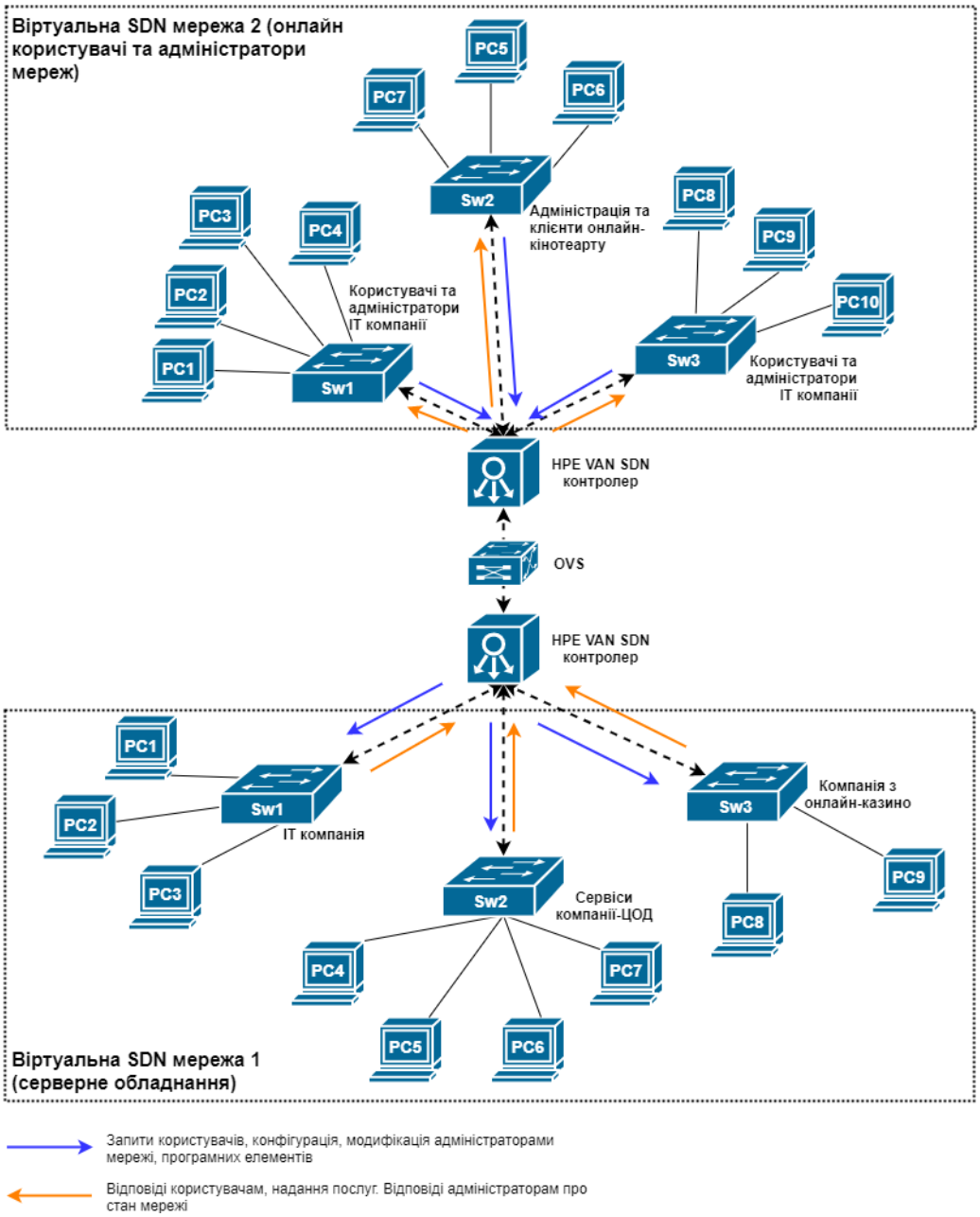


Рисунок 4.1 – Структура макету, який буде спроектовано до Mininet

Слід зазначити, що адміністрація компанії дійсно підключається до мережі через комутатор. Але користувачі – ні, вони отримують доступ до ресурсу, або контенту через мережу інтернет. Але через обмеження і неможливість це відобразити на досліджуваній платформі – користувачі компаній також будуть під'єднані через комутатори.

В даному макеті комутатори не з'єднані між собою, але якби вони з'єднувались між собою і наприклад створювали б петлю (зв'язки замикались) – контролер би не відпрацьовував. Це така особливість HP VAN

SDN контролера, що він не в змозі працювати в мережах з замкненими мережевими з’єднаннями.

Тому слід сказати декілька слів про контролер, який був вибраний для дослідження, перш ніж почати проектування мережі. HP VAN SDN контролер розвивався паралельно з розвитком SDN мереж, був опублікований у вільний доступ у 2013 році, зараз знаходиться в вільному доступі у пробній (free-trial) версії. Яка накладає на нього певний ряд обмежень, аби контролер використовувався сугубо в наукових або ж академічних цілях. Контролер має назву: “HP VAN SDN Controller”, яка розшифровується як: “HP Virtual Application Networks SDN Controller”, тобто HP контролер програмно-конфігурованих мереж з віртуальними додатками мереж. Розроблений даний віртуальний контролер вендором Hewlett Packard Enterprise (HPE) та активно покращується і підтримується розробниками.

Було вирішено вибрати саме цей контролер, оскільки вже довелось працювати з POX та Floodlight контролерами, а про дану модель контролеру мало де сказано. Тому цікаво дослідити його можливості і зробити щодо нього певні висновки.

Що ж це за контролер та які функції він надає? Розглянемо далі.

Контролер HPE VAN SDN забезпечує централізовану точку доступу в нашому випадку програмно-конфігурованій мережі з підтримкою OpenFlow, спрощуючи управління, конфігурації та моніторингу у мережі. Даний контролер, як і більшість контролерів, забезпечує конфігурацію мережевих пристроїв згідно мережевих політик, які має в наявності. А також забезпечує надання мережевих послуг нового покоління на основі додатків.

Розробники акцентують увагу на тому, що в архітектурі Hewlett Packard Enterprise Software Defined Networking (SDN) управління та мережеві площини мережі відокремлені одна від одної, централізуючи мережевий інтелект та абстрагуючи основну мережеву інфраструктуру від програм.

Програмне забезпечення контролера безпосередньо підтримує роботу як фізичних так і віртуальних комутаторів через стандартний протокол OpenFlow. Мережеві порти та топологія мережі виглядають “прозоро” для контролера, що дозволяє полегшити централізоване адміністрування політик та забезпечити більш ефективний вибір шляхів на основі динамічного уявлення про мережу. Це суттєво спрощує організацію передачі трафіку із кількома клієнтами (в нашому випадку компанії, що орендують сервера) та застосування мережевих політик як для користувачів, так і для серверів.

Контролер HPE VAN SDN може використовуватись для роботи в різних обчислювальних середовищах, такі як: кампус, центр обробки даних, приватна хмара та загальнодоступна хмара.

Особливості контролера HPE VAN SDN:

- це платформа корпоративного типу для реалізації та впровадження великої кількості мережевих інновацій;
- підтримка та відповідність протоколам OpenFlow версії 1.0 та 1.3;
- підтримка функцій мережевої віртуалізації, забезпечення безпеки, а також функції авторизації, за допомогою яких можливо керувати підключеними сторонніми SDN додатками до контролера;
- наявність вбудованих в контролер програм, що надають загальні мережеві послуги;
- підтримка комутаторів від Hewlett Packard Enterprise та H3C OpenFlow;
- відкриті API дають змогу розробникам SDN додатків пропонувати або ж надавати інноваційні рішення, які можуть одразу зв'язати бізнес-вимоги з мережевою інфраструктурою, використовуючи або власні програми Java, або загальні інтерфейси керування RESTful API;
- підтримка інтеграції з HPE Intelligent Management Center (IMC). HPE IMC в свою чергу забезпечує повноцінне управління роботою додатків контролера, покращує звітування мережевих пристроїв та візуалізацію топології мережі SDN. Візуалізація топології мережі необхідна деяким додаткам для того, аби створювати мережеві політики, орієнтуючись на поточну топологію.

Під час досліджень було помічено, що вендор HP, також зробив вагомий вклад у розробку SDN мереж та приймав активну участь у розробці SDN, коли дані мережі почали тільки розвиватися. Вендор, під час дослідження мереж, розробив 9 нових моделей комутаторів, що могли підтримувати і взаємодіяти за протоколом OpenFlow.

Наразі, майже уся лінійка комутаторів та маршрутизації, що є у HP, підтримує протокол OpenFlow. Тому мережеві пристрої можуть застосовуватись як у традиційних мережах (керування та налаштування на кожному пристрої окремо), так і у SDN мережах, відповідно.

Отже, даний контролер в нас буде віртуальним і буде розташовуватись на окремій віртуальній машині з операційною системою Linux Debian.

Тепер розглянемо які компоненти нам необхідні для створення даного мережевого макету та їх призначення:

- *Комп'ютер з підтримкою віртуалізації*. Це може бути персональний комп'ютер або ноутбук з операційною системою та з підтримкою

апаратної віртуалізації (має мати процесор з підтримкою віртуалізації). Це напевне банально, але даний хост також має бути оснащений мережевою картою та оновленими драйверами під неї.

Під *драйвером* розуміється програмне забезпечення, за допомогою якого операційна система може взаємодіяти (отримує доступ на використання) з апаратною частиною пристрою. Зараз операційні системи здібні самостійно, при підключенні нового периферійного або апаратного обладнання, шукати та автоматично оновлювати драйвери. Більшість розробників апаратного обладнання оновлюють драйвери для вирішення проблем або впровадження нових функцій. Користувачі після оновлення драйвера можуть або власноруч завантажити та оновити драйвер, з офіційного сайту виробника, або ж програма виробника апаратного обладнання (наприклад програма керування відеокартою або звуковою картою) запропонує автоматично оновити драйвер.

Слід зазначити, що віртуалізація (функція віртуалізації) може бути вимкнена по-замовчуванню на комп'ютері. Для ввімкнення даної технології слід вимкнути, або перезавантажити комп'ютер і під час повторного включення натиснути кнопку входу до BIOS (зазвичай це кнопка "F2" на клавіатурі). Після входу до середовища BIOS слід перейти на вкладку "Додаткових налаштувань" (Advanced), де вибрати пункт "Конфігурація процесора" (CPU Configuration), і у відкритому списку знайти пункт і активувати його:

для процесорів від Intel - Intel Virtualization Technology;

для процесорів від AMD - SVM Mode;

Після включення технології віртуалізації необхідно вийти з BIOS зі збереженням змін, це робиться гарячою клавішею на клавіатурі "F4". Після завантаження системи з'являється змога використовувати програми, що потребують технологію віртуалізації.

BIOS – це так зване середовище, але частіше називають мікропрограмою, яка вбудована в чіп материнської плати. В даному середовищі можна налаштовувати різноманітні параметри апаратних компонентів, що під'єднані до материнської плати комп'ютера.

- *Програмний гіпервізор VirtualBox*. Це програмний додаток, розроблений компанією Oracle для створення та керування віртуальними машинами. Даний гіпервізор являється гіпервізором на базі основної операційної системи.

Гіпервізором виступає програма, яка запускає на комп'ютері-хості (в даному випадку від хостом розуміється головний комп'ютер) паралельно декілька віртуальних машин з власною операційною системою. Дані віртуальні машини повністю ізольовані, це зроблено шляхом розподілення фізичних та логічних пристроїв віртуальних машин.

Гіпервізор виступає в ролі середовища з власним ядром та набором інструментів, які використовуються для керування віртуальними вузлами. Керує він наступним чином: створює і запускає віртуальні машини, використовуючи образи операційних систем, які попередньо завантажені на хост, та керує цими віртуальними машинами. Гіпервізор віртуалізує усі апаратні компоненти, такі як: процесор, ядра процесора, оперативну пам'ять, інтерфейси (мережеві та периферійні). Всі ці системні апаратні параметри займаються з фізичного хоста. Також гіпервізор підтримує інтерфейс паравіртуалізації.

Гіпервізори також розділяють на наступні типи:

Автономні гіпервізори. Такі програми має в своєму арсеналі вбудовані драйвери різноманітних пристроїв, це робить їх незалежними від операційної системи на хостовому комп'ютері. Такі гіпервізори частіше за все використовують на реальному обладнанні.

Гіпервізори на базі основної операційної системи (під основною операційною системою розуміється операційна система, що працює на хості, тобто на основному комп'ютері, на якому встановлено гіпервізор). Дані програми працюють в одному просторі з ядрами основного хоста. Всі процеси використовуються на фізичному процесорі але доступ до периферійних пристроїв здійснюється через інший компонент.

Гібридні гіпервізори. Такі гіпервізори складаються з двох частин: тонкого гіпервізора, який керує процесором та пам'яттю, та спеціальної операційної системи, робота якої керується гіпервізором. Саме через спеціальну сервісну операційну систему гостьові операційні системи можуть отримувати доступ до фізичного пристрою.

- Образ віртуальної машини на базі операційної системи Linux Ubuntu 14.04 зі встановленим програмним продуктом Mininet;
- Образ віртуального контролеру HPE VAN SDN (віртуальна машина зі встановленим контролером на базі операційної системи Linux);
- Програмні продукти: Bitvise SSH Client та Putty, які виступають реальними продуктами для віддаленого підключення та віддаленого доступу до файлового сховища.

Тепер перейдемо до встановлення вищезазначених компонентів.

Для встановлення програмних продуктів VirtualBox, Bitvise SSH Client та Putty слід завантажити установочний файл та пройти процедуру встановлення додатку. Після чого додатки будуть готові до роботи.

Щодо віртуальної машини з продуктом Mininet. Спочатку слід завантажити з офіційного сайту Mininet образ віртуальної машини з вбудованим Mininet.

Ця віртуальна машина являється найкращим варіантом для створення та дослідження віртуальних мереж. Графічний інтерфейс максимально простий у даної машини, весь графічний інтерфейс – термінал/консоль операційної системи Linux. Дійсно не так зручно керувати файлами як на повноцінній операційній системі, але для цього існує ряд команд і також можна використати програмне забезпечення Bitvise SSH Client, яке дасть змогу отримати доступ до файлів на віртуальному вузлу.

Після завантаження образу слід зайти до встановленого вже програмного забезпечення VirtualBox і вибрати наступні вкладки: *Файл – Імпортувати образ віртуальної машини*. Після чого вибирається завантажений раніше образ. Далі вибирається необхідна кількість ядр процесору та оперативної пам’яті, яку буде виділяти апаратна частина хосту на цю віртуальну машину. Рекомендовано виділяти на дану віртуальну машину 1-2 ядра процесора та мінімум 1 гігабайт оперативної пам'яті. Після вибору параметрів, протягом декількох хвилин буде відбуватись процес імпорту віртуальної машини до гіпервізора.

Віртуальна машина з контролером HPE VAN SDN також завантажується спочатку у вигляді образу з офіційного сайту розробника. Після чого виконується ідентичний імпорт віртуальної машини до програмного забезпечення VirtualBox. Для даної віртуальної машини рекомендовано виділяти 2-4 ядра процесора, а також 2-4 гігабайти оперативної пам’яті. Також при імпорті слід у пункті “MAC Address Policy” вибрати: “Include all network adapter MAC addresses”, що дасть змогу коректно працювати мережевим адаптерам на віртуальній машині.

Розглянемо та нагадаємо призначення усіх вищезгаданих елементів нашого макету мережі.

Програмне забезпечення VirtualBox виступає в ролі гіпервізора, його задачі це створити віртуальну машину, запустити її та підтримувати її роботу. Платформа Mininet буде виступати у якості віртуального стенду для створення та запуску віртуальної мережі, а також для аналізу параметрів у мережі. На віртуальній машині з Mininet також вбудовано програмне забезпечення Wireshark, яке виступає в ролі інструменту для відстеження

трафіку по мережевим інтерфейсам і дозволяє нам аналізувати вхідний/вихідний трафік. Контролер HPE VAN SDN підключений до мережі на Mininet та керує роботою цієї мережі. Додатки Bitvise SSH Client та Putty підключаються віддалено до віртуального вузла. Однак Bitvise SSH Client ще має доступ до файлової системи пристрою, до якого приєднався. А Putty підключається лише до терміналу/консолі підключеного пристрою.

В нас тепер присутні усі необхідні встановлені компоненти. Але слід їх ще налаштувати, для початку роботи.

Для цього нам слід налаштувати мережеві адаптери на віртуальних машинах з Mininet та HPE VAN контролером. Для цього в гіпервізорі VirtualBox заходимо до налаштувань обох віртуальних машин і у вкладці “Мережа” налаштовуємо мережевий адаптер, підключаючи його до Проміжного адаптеру. Назва адаптеру має бути вибрана під назву мережі оператора. Даний крок дасть нам змогу отримати кожній віртуальній машині власну унікальну логічну адресу, разом з цим з’являється доступ до мережі Інтернет. За допомогою цього адаптеру, віртуальні машини можуть спілкуватися один з одним, використовуючи відповідні логічні (ip) адреси. Також, використовуючи логічну адресу, програми для віддаленого керування, на кшталт Putty, можуть виконувати віддалене підключення до пристрою з відповідною адресою.

Тепер, коли в нас встановлено необхідне програмне забезпечення, віртуальні машини налаштовані, ми можемо приступати до досліджень. Для поступового розуміння роботи мережі та підходу до створення нашого головного макету, буде проведено ряд побудов мереж з поступовим додаванням певних елементів чи функцій.

Спочатку побудуємо найпростішу SDN мережу на базі платформи Mininet, для цього потрібно мати встановлену віртуальну машину з Mininet у VirtualBox. Якщо умова виконана (ми це виконали в даному розділі вище) – необхідно проробити наступні кроки:

1) Запустити віртуальний вузол та виконати авторизацію;

2) Прописати команду запуску найпростішої мережі на Mininet;

3) Дочекатися запуску та роботи мережі;

4) Перевірити працездатність мережі відповідним системним інструментом.

Спочатку запускаємо віртуальну машину з вбудованим Mininet через програмний гіпервізор VirtualBox. Після завантаження системи ми

потрапляємо на етап авторизації в системі. Тут ми прописуємо логін та пароль: “*mininet*”. І входимо до системи.

Після чого прописуємо команду для створення мережі найпростішої (мінімальної – minimal) топології:

sudo mn – де, sudo – надає користувачеві права суперкористувача (адміністратора), mn – створює та запускає найпростішу топологію мережі.

Далі слід почекати декілька секунд для запуску та конфігурації віртуальної мережі.

Після запуску мережі перевіримо її працездатність запуском інструмента “pingall”. Для цього в терміналі прописуємо команду: *pingall*. Для отримання більш детальної інформації після виконання команди можна запустити дещо модифіковану команду, яка виглядає: *pingallful*.

Якщо після виконання команди в результаті ми маємо “0% dropped” – це означає, що всі перевірочні пакети були доставлені до отримувача. Отже мережа працює. Якщо ж в результаті маємо якийсь відсоток втрат – це може означати, що якісь вузли в мережі недоступні. Але результат з втратами може також траплятися, коли одразу після запуску мережі запускається даний інструмент. Відповідно, в такому випадку, мережеві елементи ще не встигли налаштувати власну конфігурацію для роботи в мережі.

Після виконання роботи слід вимкнути мережу. Це робиться командою: *exit*.

Тепер наблизимося трохи ближче до нашої мети і запустимо цю ж саму найпростішу віртуальну мережу, під’єднавши до мережі віддалений контролер HP VAN SDN.

Для реалізації такого макету необхідні обидві наші віртуальні машини та необхідно проробити наступні дії:

1) Запустити віртуальні машини та виконати авторизацію на них;

2) Дізнатися логічні адреси адаптерів на кожній віртуальній машині;

3) Прописати команду запуску мережі на Mininet з потрібними параметрами та підключення контролеру до мережі;

4) Дочекатися запуску мережі;

5) Перевірити працездатність мережі відповідним інструментом.

Спершу запускаємо обидві віртуальні машини одночасно. Ми це можемо зробити, адже зважено розподілили ресурси на їх роботу, при їх імпортуванні до гіпервізору. Але дані налаштування, з приводу більшого

виділення ядер, оперативної пам’яті, можна виконати в будь-який час, коли віртуальна машина вимкнена. Після запуску віртуальних машин і після очікування ми потрапляємо до вікна авторизації системи, де вводимо:

Для віртуальної машини з Mininet:

Логін: *mininet*;

Пароль: *mininet*;

Для віртуальної машини з HD VAN SDN контролером:

Логін: *sdn*;

Пароль: *skyline*;

Після авторизації до системи у терміналах обидвох машин, використовуємо інструмент для перегляду підключених мережевих адаптерів на пристрої. Інструмент викликається та виконується командою: *ifconfig*.

Після виконання команди ми можемо бачити логічну адресу проміжних адаптерів обох віртуальних машин. Саме вони нам і потрібні для підключення машин між собою.

Тепер перейдемо до створення і запуску мережі. Перед створенням мережі варто очистити кеш-сховище платформи Mininet від минулого запуску. Очищення реалізовується наступною командою: *sudo mn -c*. Це робиться в цілях уникнення конфлікту між збереженими раніше файлами запуску і поточними файлами запуску.

Для початку кастомізуємо (модифікуємо) раніше використану команду необхідними параметрами для створення та запуску найпростішої віртуальної мережі з віддаленим контролером. Команда виглядатиме наступним чином:
sudo mn --controller=remote,ip=192.168.0.103 --switch=ovsk, protocol=OpenFlow13 – де,

sudo mn --controller=remote,ip=192.168.0.103 – частина, яка створює найпростішу програмно-конфігуровану мережу замінюючи контролер по-замовчуванню на віддалений контролер, логічна адреса контролеру: 192.168.0.103.

--switch=ovsk,protocol=OpenFlow13 – частина, яка каже платформі створювати мережу, використовуючи OpenFlow комутатори (OVS), які підтримують протокол OpenFlow версії 1.3.

Після виконання вищезазначеної команди почнеться створення мережі:

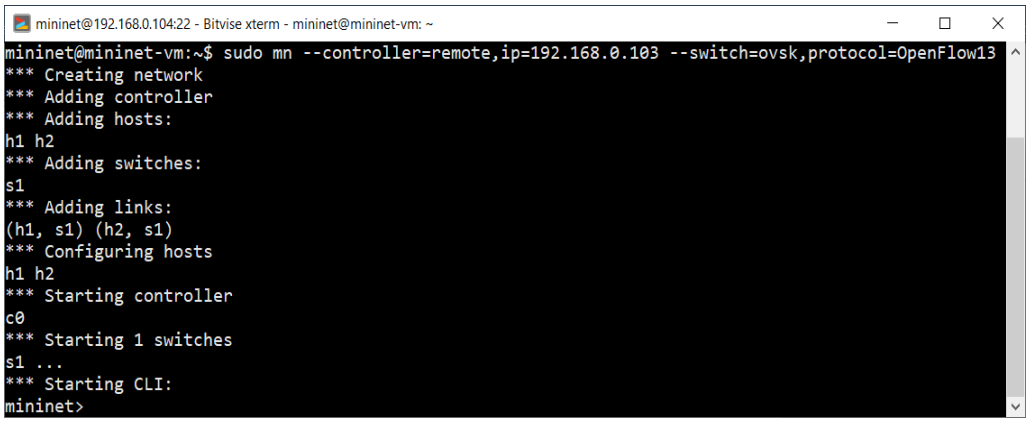


Рисунок 4.2 – Процес створення і запуску мережі через термінал Bitvise Client

Після створення мережі і налаштування компонентів мережі ми запускаємо інструмент “pingall” для перевірки працездатності мережі. Запускаємо командою: *pingall*.

Якщо після виконання команди маємо 0% втрат – отже наша мережа налаштована та готова до роботи.

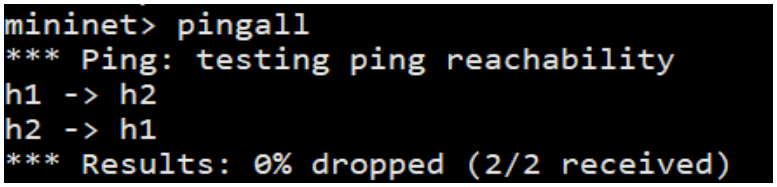


Рисунок 4.3 – Результат успішної перевірки працездатності мережі

Не забуваємо, що в нас підключений віддалений контролер. Який має власний веб-інтерфейс. Для доступу до веб-інтерфейса заходимо на основному хості до веб-браузера (бажано Google Chrome), та ввівши унікальну адресу і вказавши необхідний порт підключення, ми потрапимо до веб-інтерфейсу контролера.

Отже в адресній строчці вписуємо наступну адресу: *https:///(ip адреса віртуальної машини):8443/sdn/ui/app/index*. Де вводиться унікальна ip-адреса віртуального хосту з контролером і порт 8443 (розробники зробили веб-інтерфейс і підключили його саме до даного порту контролера).

Інші контролери, які мають власний веб-інтерфейс, можуть реалізувати його роботу на різних портах. Це все залежить сугубо від рішення розробників.

Після переходу на сторінку за даною адресою ми потрапляємо на сторінку авторизації, де авторизуємо вписуючи такі самі дані, які використовували під час авторизації до системи віртуальної машини з контролером.

Тут маємо багато вкладок, але нам потрібна “OpenFlow Topology”. Переходячи на цю вкладку, ми можемо бачити нашу топологію віртуальної мережі, яка керується даним контролером. В нашому випадку топологія виглядатиме наступним чином на графічному веб-інтерфейсі:

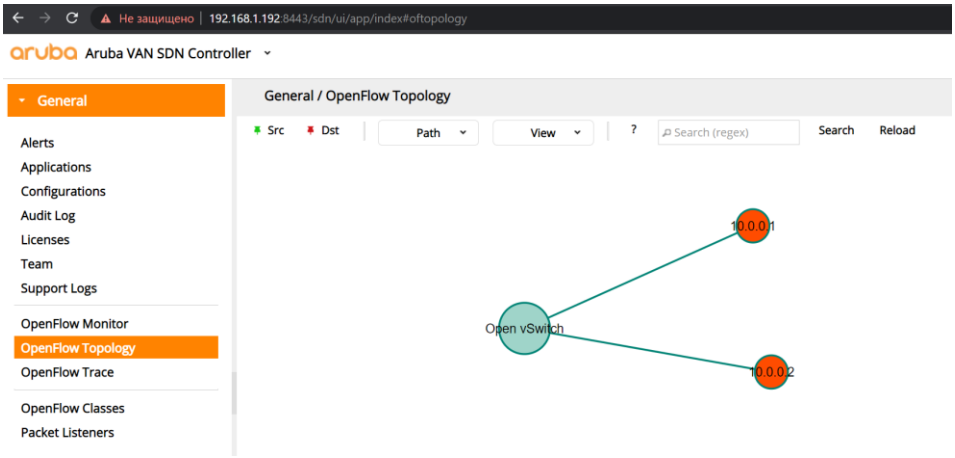


Рисунок 4.4 – Зображення топології мережі на веб-інтерфейсі контролера

Після чого не забуваємо закрити мережу, використовуючи команду: *exit*. Та виконати очищення кеш сховища Mininet, використовуючи: *sudo mn -c*.

Дана запущена мережа та взаємодія її мережевих елементів виглядає наступним чином:

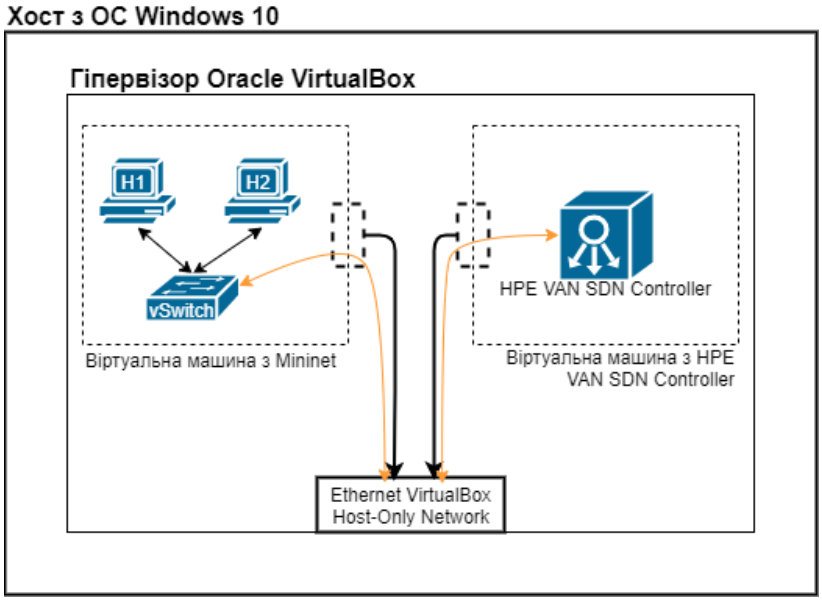


Рисунок 4.5 – Взаємодія компонентів запущеної вище мережі

4.2. Побудова першої частини кластерної мережі ЦОД з використанням платформи Mininet та віддаленого контролеру HPE VAN SDN

Тепер переходимо до головної частини, а саме: створення власної мережевої топології для віртуальної кластерної ЦОД мережі та запуск її на платформі Mininet з віддаленим контролером. Для цього нам потрібні обидві віртуальні машини, їх слід продублювати, адже в нас буде 2 мережі, котрі під'єднані до контролерів, які в свою чергу мають взаємодіяти між собою. Дублювання можна виконати в гіпервізорі VirtualBox натиснувши правою клавішею на віртуальну машину і вибрати "Клонувати", після чого слід вибрати місце для копії, її назву та клонування мережевих адаптерів.

Також ми будемо виконувати віддалене підключення до віртуальних мереж з метою отримати доступ до файлової системи і вивантажити файл з топологією до системи. Для цього будемо використовувати додаток Bitvise SSH Client, оскільки він дає змогу отримати доступ да файлової системи підключеного хосту.

Тепер побудуємо та запустимо першу частину нашої кластерної мережі ЦОД. Вона в нас відображає ділянку, де встановлені і працюють сервери компанії. Нагадаємо, що перша модель (частина) мережі набуває наступного вигляду:

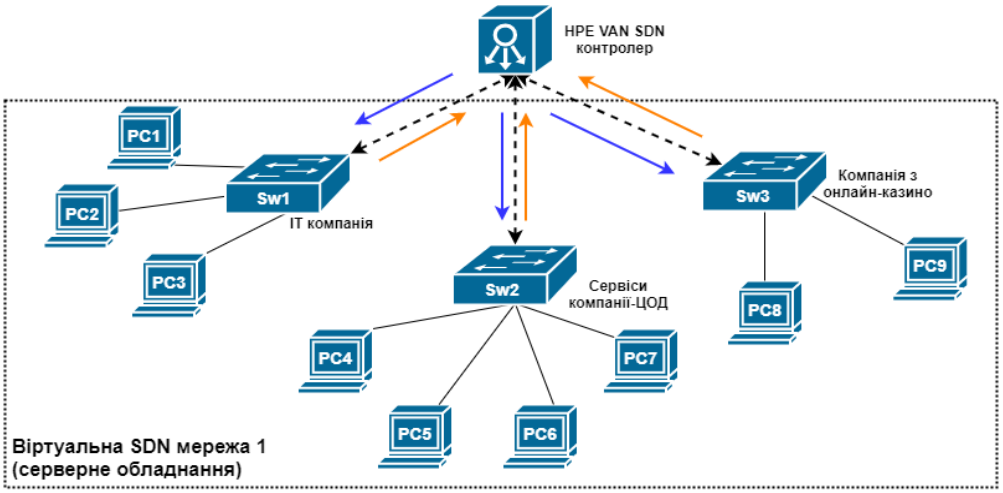


Рисунок 4.6 – Візуальний макет першої частини проектованої мережі ЦОД

Для побудови даної моделі нам слід проробити наступні кроки:

1) Запустити віртуальні машини (2 з Mininet, 2 з контролерами) та виконати авторизацію на них;

2) Дізнатися логічні адреси адаптерів на кожній віртуальній машині;

3) Виконати віддалене підключення до першої віртуальної машини з Mininet за допомогою програмного забезпечення Bitvise SSH Client;

4) Створити файл топології для мережі з серверним обладнанням, та заповнити його елементами і зв’язками згідно макету;

5) Вивантажити створену топології на систему з Mininet через файловий обмінник Bitvise SSH Client;

6) Виконати віддалене підключення до другої віртуальної машини з Mininet за допомогою програмного забезпечення Bitvise SSH Client;

7) Створити файл топології для мережі з користувачами, замовниками і адміністраторами, та заповнити його елементами і зв’язками згідно макету;

8) Вивантажити створену топології на систему з Mininet через файловий обмінник Bitvise SSH Client:

9) Прописати команду запуску мережі на першій віртуальній мережі з Mininet з потрібними параметрами та підключенням даної мережі до контролеру. Та дочекатися запуску мережі;

10) Прописати команду запуску мережі на другій віртуальній мережі з Mininet з потрібними параметрами та підключенням даної мережі до контролеру. Та дочекатися запуску мережі;

11) Перевірити працездатність мереж відповідним інструментом;

12) Виконати дослідження щодо мережевих параметрів в віртуальних мережах;

Спочатку запускаємо дві віртуальні машини з контролером. Запускаємо одразу не чотири віртуальні машини, а дві для того, щоб залишити комп'ютеру необхідну оперативну пам'ять для роботи основної операційної системи та її компонентів. Чекаємо поки віртуальні машини завантажаться. Після чого авторизуємося до системи вводячи на обидвох машинах:

Логін: *sdn*;

Пароль: *skyline*;

І переглядаємо та запам'ятовуємо унікальні адреси кожного контролера, ввівши команду: *ifconfig*.

Тепер вимикаємо віртуальні машини з контролерами та вмикаємо віртуальні машини з Mininet. Після чого виконуємо авторизацію до систем, вводячи:

Логін: *mininet*;

Пароль: *mininet*;

Після авторизації до системи у терміналах обидвох машин, використовуємо інструмент для перегляду підключених мережевих адаптерів на пристрої. Інструмент викликається командою: *ifconfig*.

Після виконання команди, ми ознайомлюємось яку логічну адресу має кожна з віртуальних машин.

Тепер здійснимо віддалене підключення до першої віртуальної машини з Mininet. Для цього відкриваємо програму Bitvise SSH Client. Переходимо до вкладки "Login". В розділі "Server" вписуємо в поле "Host" логічну адресу віртуальної машини з Mininet, також перевіряємо аби у полі "Port" було вписано значення "22" (22 порт – порт по-замовчуванню для підключення через SSH протокол).
Після цього у блоці "Authentication" в поле "Username" вводимо логін від віртуальної машини (який використовуємо під час входу до операційної системи) та у селекторі "Initial method" ми вибираємо "password" і в полі "Password" вводимо відповідно пароль від віртуальної машини (який використовуємо під час входу до операційної системи). Це робимо для того, аби при підключенні нас система одразу пропустила крізь сторінку авторизації.

Після чого натискаємо на кнопку “Log In” та чекаємо поки встановиться підключення до віртуальної машини.

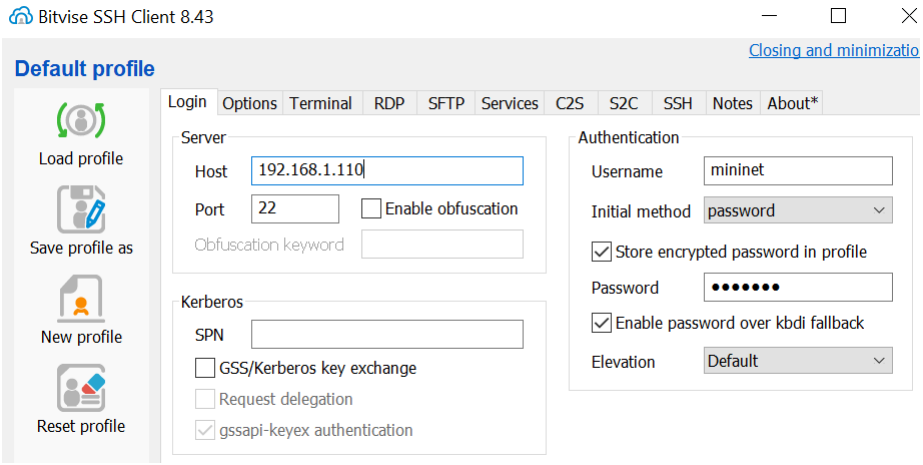


Рисунок 4.7 – Заповнення відповідних полів для віддаленого підключення

Коли відбудеться підключення – відкриються 2 вікна. Одне – термінал віртуальної машини, друге – файловий інспектор (менеджер).

Тепер створимо на основному хості файл з розширенням python (.py) де будемо описувати власну топологію мережі (макет мережі з серверним обладнанням). Після створення файлу вписуємо в нього наступний код (скрипт):

```
1. from mininet.topo import Topo
2. from mininet.link import TCLink
3.
4. class MyTopo( Topo ):
5.     "Topology for research."
6.
7.     def __init__( self ):
8.         "Creating network mockup."
9.
10.         # Initialize topology
11.         Topo.__init__( self )
12.
```

```
13.         # Add hosts and switches
14.         Host1 = self.addHost( 'h1' )
15.         Host2 = self.addHost( 'h2' )
16.         Host3 = self.addHost( 'h3' )
17.         Host4 = self.addHost( 'h4' )
18.         Host5 = self.addHost( 'h5' )
19.         Host6 = self.addHost( 'h6' )
20.         Host7 = self.addHost( 'h7' )
21.         Host8 = self.addHost( 'h8' )
22.         Host9 = self.addHost( 'h9' )
23.         Switch1 = self.addSwitch( 's1' )
24.         Switch2 = self.addSwitch( 's2' )
25.         Switch3 = self.addSwitch( 's3' )
26.
27.         # Add links
28.         self.addLink( Host1, Switch1 )
29.         self.addLink( Host2, Switch1 )
30.         self.addLink( Host3, Switch1 )
31.         self.addLink( Host4, Switch2 )
32.         self.addLink( Host5, Switch2 )
33.         self.addLink( Host6, Switch2 )
34.         self.addLink( Host7, Switch2 )
35.         self.addLink( Host8, Switch3 )
36.         self.addLink( Host9, Switch3 )
37.
38. topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Лістинг коду № 4.1 – Скрипт топології мережі першої частини макету мережі ЦОД

В даному скрипті ми можемо спостерігати, що:

У строчках 1-2 описано, що ми імпортували бібліотеки для створення власної топології та мережевих з’єднань у мережі. Кожна бібліотека містить в собі набір інструментів, які дозволяють використовувати ті чи інші

команди в файлі. Якщо необхідні додаткові функції чи команди – вони додаються прив’язкою бібліотек до файлу, як і в нашому випадку.

У 4-5 строчці ми створили власний клас для топології під назвою “MyTopo” та присвоїли опис даному класу. Важливо запам’ятати назву класу, оскільки вона далі буде фігурувати у файлі-скрипті, та під час запуску власно самої віртуальної мережі.

У 7-8 строчці ми відкриваємо (створюємо) функцію та називаємо її і описуємо відповідним коментарем.

У 11 строчці іде ініціалізація топології. Тобто ми вказали на те, що ця топологія буде локальна (буде працювати лише в межах даної віртуальної машини), оскільки при побудові топології типу: *net* виникали певні труднощі та нюанси.

У 14-15 строчках додались мережеві елементи до мережі (віртуальні комутатори та хости). Слід нагадати, що елементи ці локальні, будуть взаємодіяти лише з елементами, в рамках платформи Mininet. Вони стають локальними за допомогою приставки “self”. В сумі у нас вийшло 5 віртуальних хостів та 2 віртуальних комутатора. Схема з’єднань та взаємодії мережевих елементів зображена нижче (Рис.).

У 28-36 строчках створились мережеві з'єднання. Мережеві елементи були пов'язані один з одним мережевим каналом зв'язку.

У 38 строчці був створений ідентифікатор даної топології. Тут ми використали раніше нами створений клас топології.

Коли в нас налаштована топологія мережі, ми зберігаємо файл з топологією та в Bitvise SSH Client копіюємо його до файлового сховища віртуальної машини з Mininet. Після чого в терміналі даної віртуальної машини переходимо до папки, в якій знаходиться файл (слід це робити, якщо файл з топологією знаходиться не в “домашній” папці), наприклад якщо файл у папці custom, то необхідно прописати: *cd custom*.

Саме зараз слід ввімкнути одну віртуальну машину з контролером, який буде підключатись до даної мережі, після чого авторизуватись до системи з контролером, для початку його роботи.

Тепер можемо відключатися від першої віртуальної машини у Bitvise SSH Client. І в терміналі даної віртуальної машини з Mininet прописуємо команду для запуску нашої користувацької віртуальної мережі та підключення до неї HP VAN SDN контролера. Зробимо це наступною командою:

sudo mn --controller=remote,ip=192.168.0.103 --switch=ovsk,protocol=OpenFlow13 --custom=research.py --topo=mytopo

де, sudo mn --controller=remote,ip=192.168.0.103 – частина, яка створює найпростішу програмно-конфігуровану мережу, замінюючи контролер по-замовчуванню на віддалений (зовнішній) контролер, підключатись до якого буде за вказаною адресою: 192.168.0.103.

--switch=ovsk,protocol=OpenFlow13 – параметри, які вказують платформі створювати мережу, використовуючи OpenFlow комутатори (OVS), які підтримують протокол OpenFlow версії 1.3.

--custom=research.py --topo=mytopo – частина, яка сповіщає Mininet про те, що буде створюватись, запускатися користувацька (custom) топологія мережі з файлу “research.py”, ідентифікатор мережі – “mytopo” (ми це прописували в кінці файлу топології).

Підключення на даному контролері приймаються на 6633 порті. Це можна також вказати в параметрах запуску мережі, ввівши: *sudo mn --controller=remote,ip=192.168.0.103,port=6633*. Але частіше за все, навіть якщо не вписати порт, мережа самостійно знайде і підключиться на необхідний порт.

Після декількох секунд завантаження, наша віртуальна мережа з користувацькою топологією запущена та готова до роботи.

Слід перевірити доступність відповідних мережевих вузлів, запуском команди: *pingall*.

Після виконання команди маємо наступний результат:

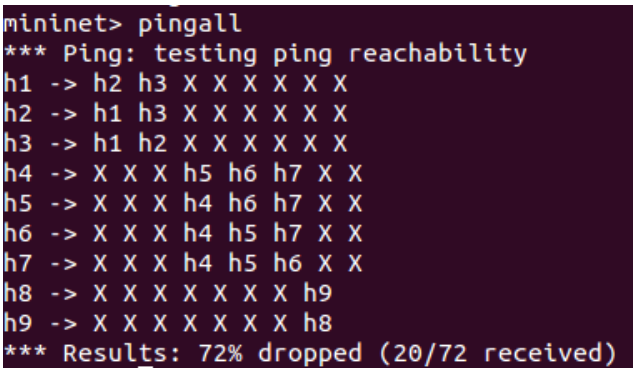


Рисунок 4.8 – Результати доступності мережевих елементів в першій частині мережі

Як і видно зі структури нижче, мережеві вузли мають змогу спілкуватися лише з тими вузлами, що підключені до одного й того ж самого комутатора. Так і має бути, насправді, адже це різні компанії і у кожного власні користувачі, сервіси, послуги, продукти. Отже працює мережа так, як і планувалось.

Загальна схема системи, мережі та взаємодія їх компонентів зображена на рисунку нижче:

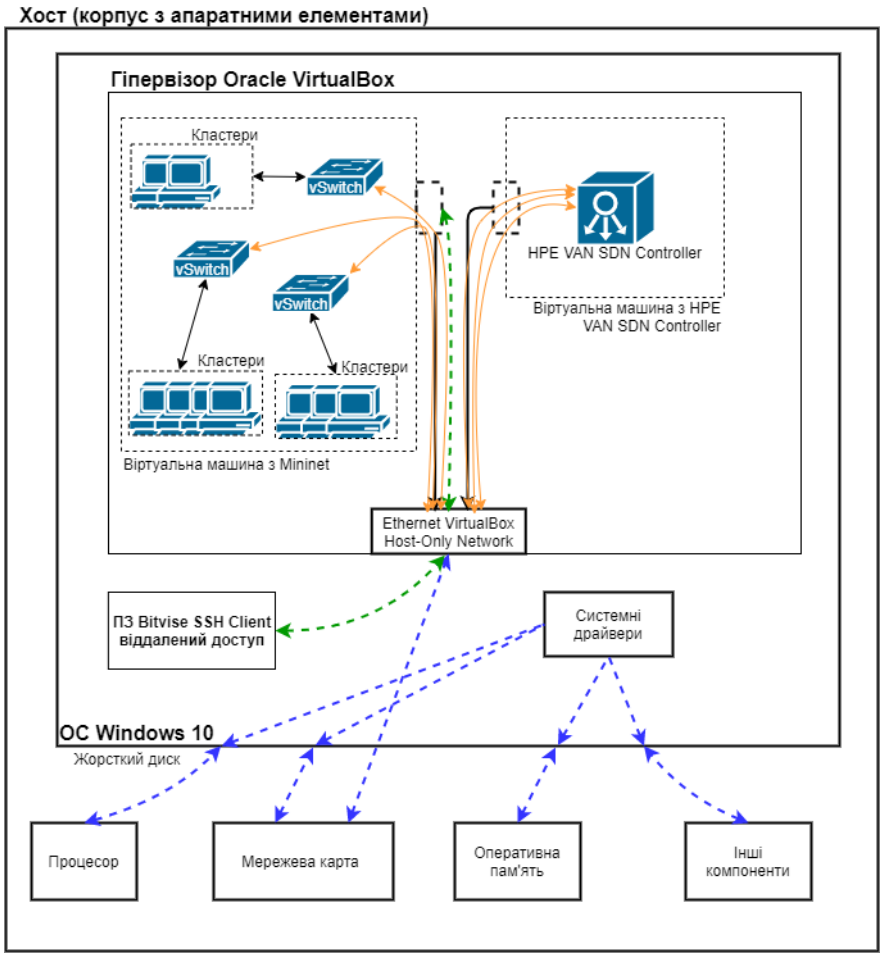


Рисунок 4.9 – Взаємодія програмних та апаратних компонентів створеної вище мережі

4.3. Побудова другої частини кластерної мережі ЦОД з використанням платформи Mininet та віддаленого контролеру HPE VAN SDN

Тепер ідентичним способом ми маємо налаштувати і запустити мережу другої частини нашого макету на другій віртуальній машині з контролером.

Макет другої частини мережі, яка призначена для користувачів та адміністрації ЦОД, виглядатиме наступним чином:

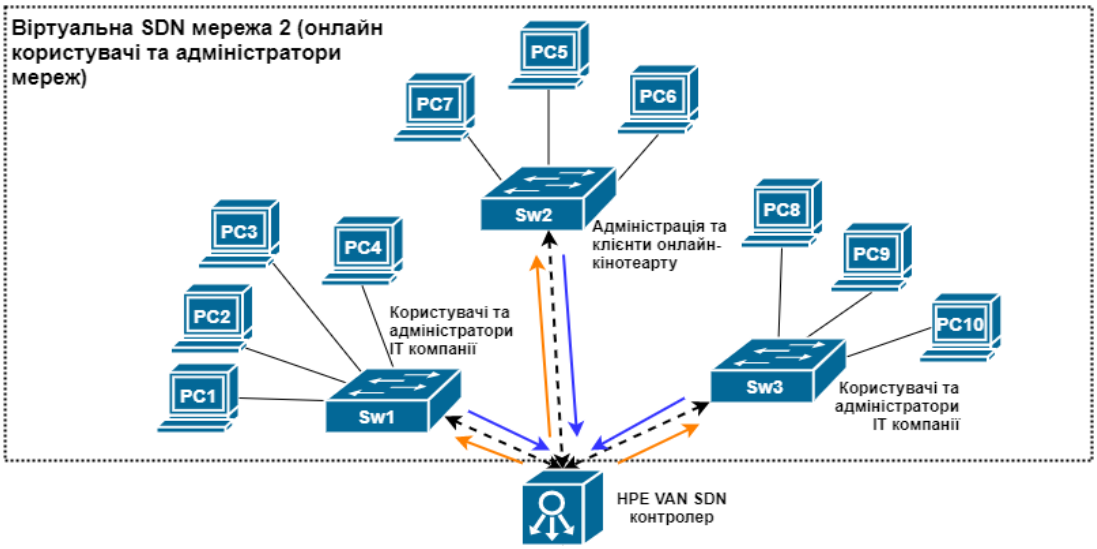


Рисунок 4.10 – Візуальний макет другої частини мережі ЦОД

Для початку слід вимкнути обидві віртуальні машини, що працювали до цього над створенням першої частини макету. Після вимкнення слід ввімкнути другу віртуальну машину з Mininet і одразу ввімкнемо віртуальну машину з другим контролером. Після включення і авторизації до машин, так само підключаємося віддалено через Bitvise SSH Client до віртуальної машини з Mininet, використовуючи для підключення унікальну логічну адресу вже цієї машини.

Після чого створюємо на основному хості ще один файл, можна попередній скопіювати та відредагувати. Код топології мережі, після редагування, набуватиме наступного вигляду:

```
1. from mininet.topo import Topo
2. from mininet.link import TCLink
3.
4. class MyTopo( Topo ):
5.     "Topology for research."
6.
7.     def __init__( self ):
```

```
8.         "Creating network mockup."
9.
10.         # Initialize topology
11.         Topo.__init__( self )
12.
13.         # Add hosts and switches
14.         Host1 = self.addHost( 'h1' )
15.         Host2 = self.addHost( 'h2' )
16.         Host3 = self.addHost( 'h3' )
17.         Host4 = self.addHost( 'h4' )
18.         Host5 = self.addHost( 'h5' )
19.         Host6 = self.addHost( 'h6' )
20.         Host7 = self.addHost( 'h7' )
21.         Host8 = self.addHost( 'h8' )
22.         Host9 = self.addHost( 'h9' )
23.         Host10 = self.addHost( 'h10' )
24.         Switch1 = self.addSwitch( 's1' )
25.         Switch2 = self.addSwitch( 's2' )
26.         Switch3 = self.addSwitch( 's3' )
27.
28.         # Add links
29.         self.addLink( Host1, Switch1 )
30.         self.addLink( Host2, Switch1 )
31.         self.addLink( Host3, Switch1 )
32.         self.addLink( Host4, Switch2 )
33.         self.addLink( Host5, Switch2 )
34.         self.addLink( Host6, Switch2 )
35.         self.addLink( Host7, Switch2 )
36.         self.addLink( Host8, Switch3 )
37.         self.addLink( Host9, Switch3 )
38.         self.addLink( Host10, Switch3 )
39.
40. topos = { 'mytopo': ( lambda: MyTopo() ) }
```

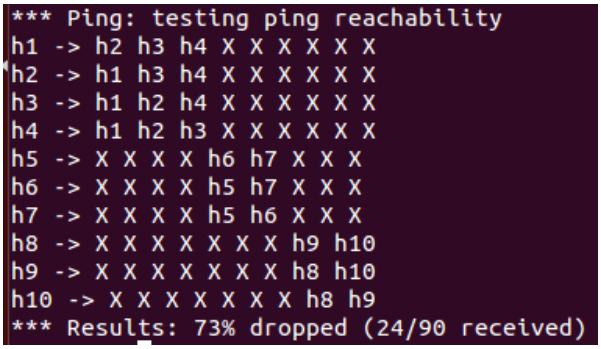
Лістинг коду №4.2 – Скрипт топології мережі другої частини макету мережі ЦОД

І далі, через Bitvise SSH Client копіюємо файл з даною топологією на віртуальну машину з Mininet.

У терміналі цієї машини переходимо до папки, де зберігається файл з топологією мережі. І виконуємо запуск топології, прописавши: *sudo mn --controller=remote,ip=192.168.0.110 --switch=ovsk,protocol=OpenFlow13 --custom=research.py --topo=mytopo*

де, в даному випадку, логічна адреса контролера – 192.168.0.110.

Після запуску мережі, запустимо і перевіримо доступність мережевих елементів, згідно макету мережі, інструментом pingall. Отримаємо наступні результати:



Рисунки 4.11 – Результати доступності мережевих елементів в другій частині мережі

Тут бачимо схожу ситуацію, як і у випадку з першою частиною мережі. Всі мережеві вузли можуть спілкуватися лише з вузлами, які знаходяться в складі одного кластеру (з вузлами, що підключені до одного комутатора). Отже мережа працює як і планувалося.

Після створення та запуску необхідних частин кластерної мережі ЦОД, було б непогано нагадати, що деякі віртуальні контролери мають власний графічний веб-інтерфейс. HP VAN SDN контролер входить в групу таких контролерів. Він має власний веб-інтерфейс за відповідною адресою: https://192.168.0.110:8443/sdn/ui/app/index. З адреси можемо бачити, що цей інтерфейс працює на 8443 порті контролера і включає в себе також унікальну логічну адресу контролера (в даному випадку 192.168.0.110). Потрапляючи

на даний домен, браузер видає сповіщення, що підключення не захищене, ми розуміємо це і натискаємо продовжити. Після цього ми потрапляємо безпосередньо до самого графічного інтерфейсу. У графічному інтерфейсі нас, як не дуже досвідчених в цьому комутаторі користувачів, цікавить 3 вкладки: OpenFlow Monitor, OpenFlow Topology та OpenFlow Trace.

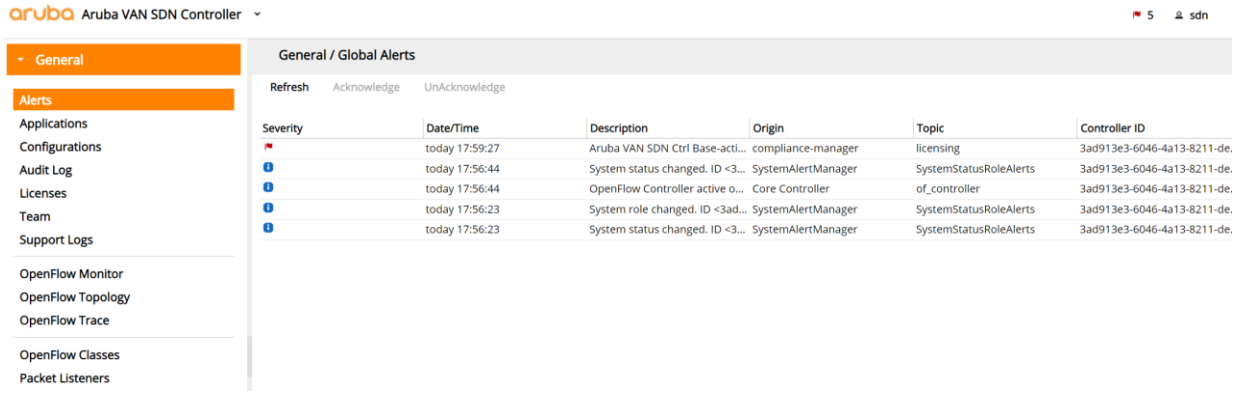


Рисунок 4.12 – Графічний веб-інтерфейс HP VAN SDN контролеру

На вкладці OpenFlow Monitor ми маємо змогу подивитись інформацію про мережеві пристрої (комутатори). А саме, ми можемо переглянути які мережеві пристрої підключені до контролеру, а також іншу інформацію про них (включаючи їх унікальну адресу):

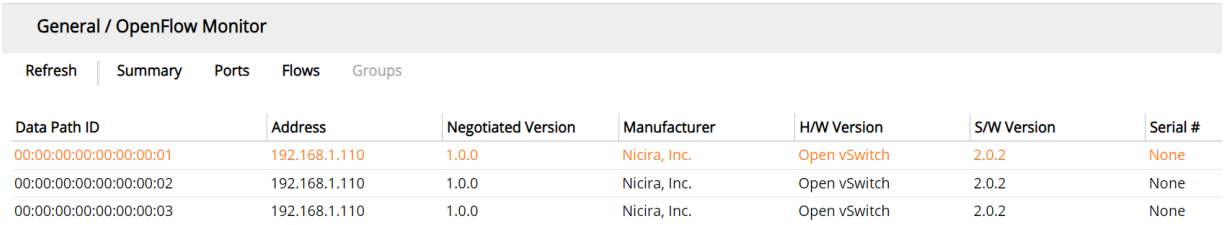


Рисунок 4.13 – Інформація про наявні комутуючі пристрої в мережі

Перейшовши до мережевого пристрою, з’являється змога переглянути більш детальну інформацію про пристрій, а саме: загальну інформацію, порти (інформація про порти, що зайняті на пристрої) та потоки (різні політики для роботи з трафіком).

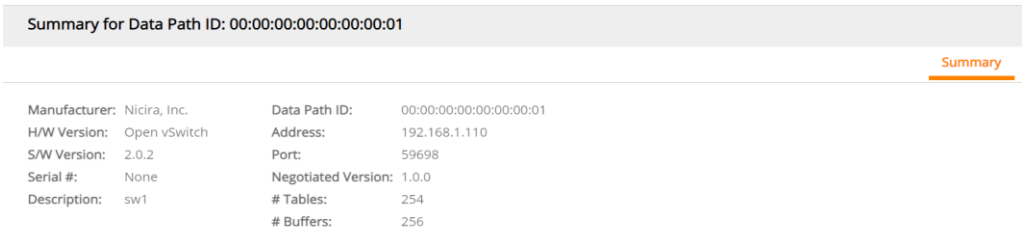


Рисунок 4.13 – Загальна інформація про мережевий пристрій

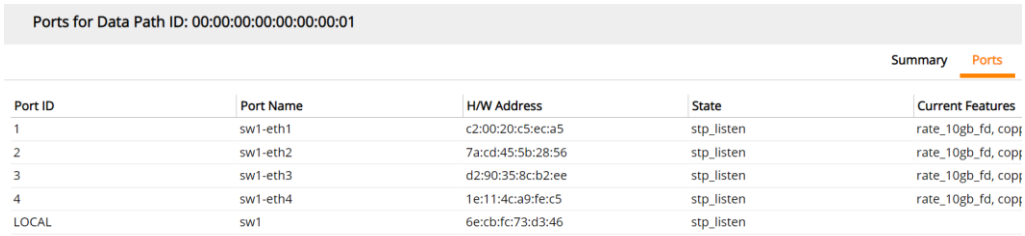


Рисунок 4.14 – Інформація про порти пристрою комутації

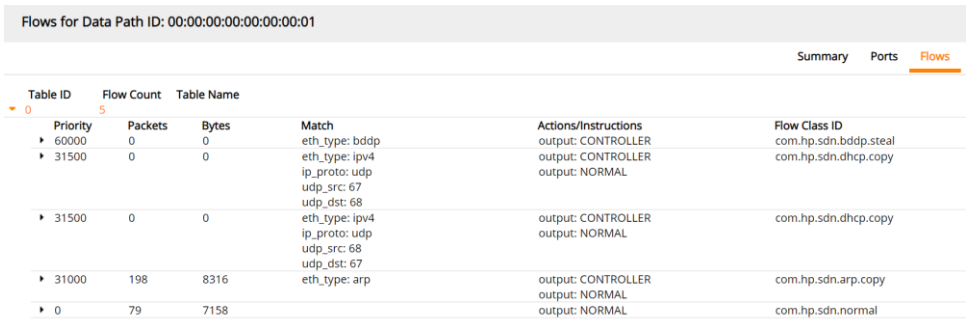


Рисунок 4.15 – Інформація про політики потоків на пристрої комутації

Що важливо, коли ми говорили про таблиці потоків на комутаторі, то зазначали, що звірка починається з таблиці 0. Тут так само бачимо, що усі політики розташовані в таблиці 0, до неї і надходить і обробляється весь вхідний трафік, що поступає на комутатор.

На вкладці OpenFlow Topology ми можемо побачити візуальний вид топології мережі, що підключена і керується контролером.

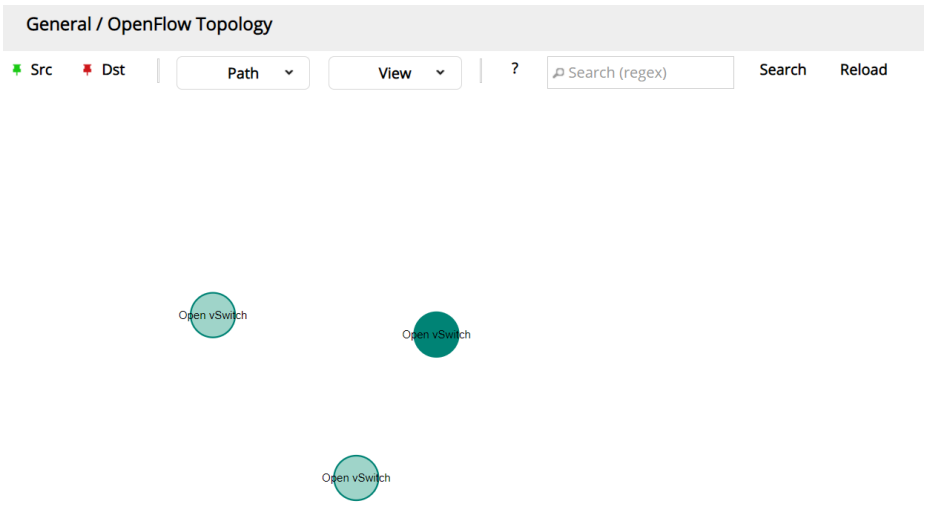


Рисунок 4.16 – Візуальне зображення топології у графічному інтерфейсі контролеру

Бачимо, що топологія відображається не дуже коректно. Це скоріше всього відбувається через відсутність мережевих з'єднань між комутаторами. Якщо ми додамо зв'язки між комутаторами, то на інтерфейсі ми лише побачимо нескінчену загрузку топології, це скоріш за все, причинено великою кількістю мережевих вузлів в мережі.

Але нам все одно такий тип з'єднання нам не підходить, оскільки в нас мережа ЦОД і комутатори тут не поєднані між собою.

На вкладці OpenFlow Trace ми можемо спостерігати за трафіком, що передається по мережі, проходячи через даний мережевий елемент. Наприклад запустимо перевірочні пакети з хоста 1 на хост 3 по мережі серверу. Уявимо ситуацію, що довго не поступало відповіді від серверу на хості 3 і сервер на хості 1 відправляє пакет, тим самим питаючи: "З тобою все гаразд?". Відправимо один пакет, команда набуватиме вигляду: *h1 ping h3*. На інтерфейсі активуємо відстеження трафіку і це виглядає наступним чином:

Time	Event	Data Path ID	Message
19:49:37.030	Tx	00:00:00:00:00:00:00:03	{ofm:[V_1_0,ECHO_REPLY,8,0]}
19:49:37.032	Tx	00:00:00:00:00:00:00:01	{ofm:[V_1_0,ECHO_REPLY,8,0]}
19:49:37.033	Rx	00:00:00:00:00:00:00:02	{ofm:[V_1_0,ECHO_REQUEST,8,0]}
19:49:37.033	Tx	00:00:00:00:00:00:00:02	{ofm:[V_1_0,ECHO_REPLY,8,0]}
19:49:37.414	Tx	00:00:00:00:00:00:00:01	{ofm:[V_1_0,PACKET_OUT,91,12270],acts=[{Act:[OUTPUT,len=8],port=0x1(1),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.414	Tx	00:00:00:00:00:00:00:01	{ofm:[V_1_0,PACKET_OUT,91,12271],acts=[{Act:[OUTPUT,len=8],port=0x1(1),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.414	Tx	00:00:00:00:00:00:00:01	{ofm:[V_1_0,PACKET_OUT,91,12272],acts=[{Act:[OUTPUT,len=8],port=0x2(2),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.414	Tx	00:00:00:00:00:00:00:01	{ofm:[V_1_0,PACKET_OUT,91,12273],acts=[{Act:[OUTPUT,len=8],port=0x2(2),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.414	Tx	00:00:00:00:00:00:00:01	{ofm:[V_1_0,PACKET_OUT,91,12274],acts=[{Act:[OUTPUT,len=8],port=0x3(3),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.414	Tx	00:00:00:00:00:00:00:01	{ofm:[V_1_0,PACKET_OUT,91,12275],acts=[{Act:[OUTPUT,len=8],port=0x3(3),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.414	Tx	00:00:00:00:00:00:00:01	{ofm:[V_1_0,PACKET_OUT,91,12276],acts=[{Act:[OUTPUT,len=8],port=0x4(4),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.414	Tx	00:00:00:00:00:00:00:01	{ofm:[V_1_0,PACKET_OUT,91,12277],acts=[{Act:[OUTPUT,len=8],port=0x4(4),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.415	Tx	00:00:00:00:00:00:00:02	{ofm:[V_1_0,PACKET_OUT,91,12278],acts=[{Act:[OUTPUT,len=8],port=0x1(1),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.415	Tx	00:00:00:00:00:00:00:02	{ofm:[V_1_0,PACKET_OUT,91,12279],acts=[{Act:[OUTPUT,len=8],port=0x1(1),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.415	Tx	00:00:00:00:00:00:00:02	{ofm:[V_1_0,PACKET_OUT,91,12280],acts=[{Act:[OUTPUT,len=8],port=0x2(2),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.415	Tx	00:00:00:00:00:00:00:02	{ofm:[V_1_0,PACKET_OUT,91,12281],acts=[{Act:[OUTPUT,len=8],port=0x2(2),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.415	Tx	00:00:00:00:00:00:00:02	{ofm:[V_1_0,PACKET_OUT,91,12282],acts=[{Act:[OUTPUT,len=8],port=0x3(3),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.415	Tx	00:00:00:00:00:00:00:02	{ofm:[V_1_0,PACKET_OUT,91,12283],acts=[{Act:[OUTPUT,len=8],port=0x3(3),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.415	Tx	00:00:00:00:00:00:00:03	{ofm:[V_1_0,PACKET_OUT,91,12284],acts=[{Act:[OUTPUT,len=8],port=0x1(1),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.415	Tx	00:00:00:00:00:00:00:03	{ofm:[V_1_0,PACKET_OUT,91,12285],acts=[{Act:[OUTPUT,len=8],port=0x1(1),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.415	Tx	00:00:00:00:00:00:00:03	{ofm:[V_1_0,PACKET_OUT,91,12286],acts=[{Act:[OUTPUT,len=8],port=0x2(2),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.415	Tx	00:00:00:00:00:00:00:03	{ofm:[V_1_0,PACKET_OUT,91,12287],acts=[{Act:[OUTPUT,len=8],port=0x2(2),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.415	Tx	00:00:00:00:00:00:00:03	{ofm:[V_1_0,PACKET_OUT,91,12288],acts=[{Act:[OUTPUT,len=8],port=0x3(3),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:37.416	Tx	00:00:00:00:00:00:00:03	{ofm:[V_1_0,PACKET_OUT,91,12289],acts=[{Act:[OUTPUT,len=8],port=0x3(3),maxLen=65535(NO_BUFFER)}],frameSize=67,pa...
19:49:40.572	CkPt		Recording stopped (forced)

Рисунок 4.16 – Відстеження трафіку на графічному інтерфейсі контролеру

Бачимо, що дуже багато чого відправилося за 1 пакет. Дане спостереження також можна завантажити у вигляді репорту (звіту зі спостережень) у форматі .csv файлу. Даний файл відкривається без проблем у програмі Microsoft Excel і добре підходить для аналізу трафіку.

Далі перед нами стоїть задача поєднати мережі ЦОД між собою. Для цього були зібрані найвдаліші думки, варіанти з’єднання мереж між собою та були проаналізовані. А саме були запропоновані наступні 3 основні варіанти:

1) Обидві частини мережі ЦОД підключити до одного контролера, тим самим поєднати віртуальні машини з мережами між собою. Однак з даною реалізацією можуть виникнути деякі складності, оскільки одна мережа підключається до контролеру і використовує контролер повністю в своїх цілях. З контролером схожа ситуація, коли до контролера надходить підключення – він використовує усі свої ресурси, додатки, для підтримки роботи підключеної мережі. Якщо підключити одну мережу до контролера, а потім іншу мережу до цього ж контролера – друга мережа можливо просто підключиться до контролера і перехопить підключення, тим самим перша мережа, скоріш за все втратить змогу працювати (оскільки таблиці комутації і усі конфігурації були відключені разом з контролером).
2) Створити окрему мережу NAT на гіпервізорі під віртуальні машини з частинами мережі (між двома віртуальними машинами) та підключити обидві віртуальні машини до цієї мережі через відповідні мережеві інтерфейси. Цей метод заключається у створені окремої локальної мережі з використанням гіпервізору і під’єднанні кожної віртуальної машини через відповідний NAT-адаптер. Однак при впроваджені даного варіанту підключення також можуть виникнути питання, які

пов’язані з тим, що саме віртуальні машини (операційні системи та системні інтерфейси) отримають доступ одна до одної, але це стосується лише Проміжного адаптеру, який являється адаптером гіпервізору і надає віртуальним машинам унікальні логічні адреси, відповідно до провайдера. А як ми можемо пам’ятаємо з вищезгаданого прикладу, що після запуску мережі на Mininet, усі мережеві інтерфейси на хості вимикаються і працює лише адаптер Mininet мереж, який носить адресу: 10.0.2.15. А тепер подумаємо, якщо ми запускаємо на обидвох віртуальних машинах мережі Mininet, то вони відмикають усі інтерфейси і працюють на єдиному Mininet інтерфейсі і виходить, що на обох віртуальних машинах активною являється логічна адреса: 10.0.2.15.

3) Кожну частину макету кластерної ЦОД мережі підключити до окремого контролеру і поєднати контролери між собою. Даний варіант передбачає паралельний запуск усіх віртуальних машин, що у нас застосовуються та відповідно подальше поєднання віртуальних машин з контролерами з окремою віртуальною машиною, на якій встановлений та налаштований програмний комутатор (наприклад: Open Virtual Switch від OpenStack). При правильному налаштуванні на комутаторі віртуальних мережевих портів та мостів може надатись змога напряму передавати трафік між комутаторами. Але тут існує один нюанс, який ідентичний з минулим варіантом з’єднання, що мережі працюють за однаковою логічною адресою, що може стати завадою для спільної роботи поєднаних частин макету.

Як бачимо, з усіх запланованих варіантів немає чіткого варіанту, який працював би без проблем, питань. Для вирішення або винайдення нового працюючого варіанту слід проробити багато спроб, використовуючи нові методики. Саме для цього і проводяться численні нові досліди, спроби побудови, з’єднання мереж, щоб знайти можливі варіанти взаємодії мереж, які можуть відкрити собою майбутні перспективні проекти або рішення у сфері побудови та дослідження віртуальних SDN мереж.

4.4. Висновки до четвертого розділу

Було представлено наближену до реальної ситуацію, коли існує дата центр (ЦОД), який в свою чергу здає в оренду серверне обладнання, а також надає послуги власним користувачам. Послуги такого дата центру придбали 2 різноманітних замовники під свої послуги для своїх клієнтів.

Макет такої системи був перероблений під реалізацію на платформі Mininet. А саме був розбитий на 2 мережі. Одна з яких відповідає за серверне обладнання, а інша за клієнтів та адміністрацію ЦОД. Кожна мережа будувалась на окремій віртуальній машині і підключалась до ще однієї віртуальної машини з контролером HPE VAN SDN. Кожна віртуальна машина має власну унікальну адресу, за допомогою якої вона могла підключатися до іншої машини.

Кожна з двох мереж створювалась у вигляді коду (скрипту) на мові Python відповідно структурі. У коді було описано: додання необхідних Python бібліотек, додання мережевих елементів, їх з'єднання між собою та ідентифікатори того, що мережа на базі Mininet. Після чого такий скрипт запускався командою Mininet і за допомогою спеціальних параметрів та унікальної адреси підключався до мережі віддалений контролер HPE VAN SDN. Після чого були проведені перевірки доступності елементів у мережі, для з'ясування: чи можуть вузли з одного кластеру, досягти та передати пакети на вузли в іншому кластері. В результаті отримали задовільний результат: сервери з одного кластеру мережі не могли взаємодіяти з серверами іншого кластеру. Після отриманого результату були запропоновані подальші кроки для поєднання обидвох мереж макету для подальшої їх взаємодії між собою.

Було розглянуто академічний віртуальний контролер HPE VAN SDN. Який розроблений під час розвитку мереж SDN і являється гарним прикладом контролера для використання у академічних дослідженнях SDN мереж. Такий контролер встановлюється лише у вигляді віртуальної машини. Має в наявності графічний веб-інтерфейс, на якому знаходяться усі необхідні відомості про керовану мережу, її топологію та комутуючі мережеві елементи (інформація про підключення, таблиці потоків).

ВИСНОВОК

В даній роботі було розглянуто багато питань, щодо призначення кластерних мереж, мереж центрів обробки даних їх нюанси, особливості. А також сучасні рішення щодо побудови цих мереж і об'єднання їх в єдину цілу мережу. Були досліджені принципи роботи технологій віртуалізації та кластеризації, типи їх та сфери, приклади застосування даних технологій.

Було розглянуто мережеву технологію SDN, її елементи, особливості, а також сучасні можливості її реалізації. Були розглянуті та запропоновані варіанти побудови кластерної мережі ЦОД на даній технології.

Було розглянуто та використано на практиці платформу Mininet для емуляції/імітації програмно-конфігурованих віртуальних мереж. Були зображені принципи її роботи, приклади реалізації мереж різної складності та топології. А також розглянуті її мережеві елементи, способи встановлення платформи. Було розглянуто особливості побудови мереж з залученням віртуального HP VAN SDN контролера. Була практично відображена спроба побудови макету кластерної віртуальної мережі ЦОД з віддаленим контролером мережі на базі платформи Mininet. Також були запропоновані варіанти, щодо з'єднання віртуальних мереж на базі Mininet, а також аспекти і способи їх вирішення.

Було досліджено та розглянуто на практичній реалізації віртуальний контролер HP VAN SDN. Розглянуто його переваги, недоліки, особливості та способи застосування. Було досліджено графічний веб-інтерфейс контролеру та його можливості і особливості.

Під кінець можна сказати, що майбутнє систем та мереж ЦОД суттєво залежить від віртуальних мереж, таких як SDN. А також від підтримки та розвитку технології SDN мереж в цілому, технології кластеризації у мережах та віртуалізації нових функцій, пристроїв та процесів. Так як з впровадженням нових функцій, підтримкою нових протоколів, будуть з'являтися нові можливості побудови мереж, нові способи надання послуг, нові типи послуг. А це в свою чергу відобразиться на вимоги користувачів послуг, замовників, яким потрібне нове обладнання для обслуговування користувачів. Також важливо не забувати про централізовані контролери, які в більшості випадків керують SDN мережами, їх також слід розвивати, надавати їм більше автономності, інтелекту, аби можливо було автоматизувати роботу мереж цілком. А з розробкою нових додатків та підтримкою контролерами декількох мов програмування, контролери

зможуть отримати можливість виконувати нові, більш потужні функції у мережах.

В академічних цілях побудови SDN мереж на просторах Інтернету та форумах, присвячених SDN тематиці, лідером являється програмний емулятор Mininet. За допомогою якого з'являється можливість розгорнути імітаційну модель реальної мережі на комп'ютері, сервері, конфігурувати мережу під власні вимоги, підключити реальне обладнання до мережі (комутатори, контролер), підключити різноманітні функції до мережі, використовуючи незчисленні можливості бібліотек мови Python, а також дослідити параметри роботи мережі. А за допомогою графічних інтерфейсів контролерів можливо побачити візуальну структуру топології мережі, що підключена до централізованого контролеру, а також отримати детальні відомості про мережеві компоненти в мережі (таблиці потоків на комутаторах, підключення на інтерфейсах комутатора).

В цілому, студент під час підготовки даної роботи покращив власні знання та навички в побудові та дослідженні SDN мереж. Покращив знання та навички в побудові віртуальних мереж на базі платформи Mininet. Розширив знання та навички в підключенні та використанні віртуального контролера в мережах SDN. Вивчив та засвоїв матеріал щодо кластерних мереж, мереж центру обробки даних, а також технологій кластеризації та віртуалізації у мережах. Освоїв та закріпив навички, щодо створення, побудови і запуску віртуальних кластерних мереж SDN у практичній частині роботи.

Під час підготовки дипломної роботи, студент активно брав участь у науковій активності, внаслідок чого, за результатами досліджень були зроблені дві доповіді та представлені на XV Міжнародній науково-технічної конференції "Перспективи телекомунікацій 2021". Тези доповідей опубліковані у матеріалах конференції. На час закінчення оформлення дипломної роботи, оформлено і надіслано статтю у науковий збірник «Information and Telecommunication Sciences».

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Перехід від звичайної мережі ЦОД до SDN системи / Huawei Russia / Блог компанії Huawei / https://habr.com/ru/company/huawei/blog/337918/
2. Кластер мережі – їх типи / kompkimi.ru
3. Кластерні системи / Сергій Лис / Блог компанії Parking.ru / https://habr.com/ru/company/parking-old/blog/126415/
4. Форум операційної системи Linux Ubuntu / пояснення програмних компонентів / http://manpages.ubuntu.com/manpages/trusty/man8/
5. Офіційний сайт Open Virtual Switch / Опис програмних елементів / http://www.openvswitch.org/support/dist-docs/
6. Кластерна архітектура / Комп'ютерна схемотехніка та архітектура комп'ютерів / intellect.icu
7. Мережева інфраструктура ЦОД – сучасний транспорт інформаційних потоків / tadviser.ru
8. SDN для ЦОД систем: як впровадити безпеку / CNews / giprosvjaz.by
9. Resource optimization in SDN-based inter/intra data centre networks / Rafael Montero Herrera / Doctoral thesis / Universitat Politècnica de Catalunya. Departament de Teoria del Senyal i Comunicacions / Barcelona / 2020-07-23 / https://upcommons.upc.edu/bitstream/handle/2117/328430/TRMH1de1.pdf
10. https://habr.com/ru/company/hpe/blog/255363/
11. https://techhub.hpe.com/eginfolib/networking/docs/sdn/sdnc2_6/5998-8473install/content/s_download_sw.html
12. Романов, О. І. ., Марінов, А. І. ., & Сколець, С. С. . (2021). ДОСЛІДЖЕННЯ МЕРЕЖ SDN З ВИКОРИСТАННЯМ ПЛАТФОРМИ MININET. Збірник матеріалів Міжнародної науково-технічної конференції «ПЕРСПЕКТИВИ ТЕЛЕКОМУНІКАЦІЙ», 98–101;
13. Романов, О. І. ., Марінов, А. І. ., & Сколець, С. С. . (2021). ДОСЛІДЖЕННЯ ПОКАЗНИКІВ ФУНКЦІОНУВАННЯ МЕРЕЖІ SDN З ВИКОРИСТАННЯМ MININET І FLOODLIGHT CONTROLLER. Збірник матеріалів Міжнародної науково-технічної конференції «ПЕРСПЕКТИВИ ТЕЛЕКОМУНІКАЦІЙ», 109–112.